RICE UNIVERSITY

Hermite Methods for the Simulation of Wave Propagation

by

Arturo Vargas

A Thesis Submitted
in Partial Fulfillment of the
Requirements for the Degree

Doctor of Philosophy

Approved, Thesis Committee:

Timothy Warburton, Thesis Director
John K. Costain Faculty Chair in the
College of Science
Professor, Virginia Tech

Adrianna Gillman, Chair
Assistant Professor

Matthew Knepley,
Assistant Professor

Stephen Bradshaw,
Assistant Professor

Houston, Texas

March, 2017

ABSTRACT

Hermite Methods for the Simulation of Wave Propagation

by

Arturo Vargas

Simulations of wave propagation play a crucial role in science and engineering. In applications of geophysics, they are the engine of many seismic imaging algorithms. For electrical engineers, they can be a useful tool for the design of radars and antennas. In these applications achieving high fidelity, simulations are challenging due to the inherent issues in modeling highly oscillatory waves and the associated high computational cost of high-resolution simulations. Thus the ideal numerical method should be able to capture high-frequency waves and be suitable for parallel computing.

In both seismic applications and computational electromagnetics the Yee scheme, a finite difference time domain (FDTD) method, is the method of choice for structured grids. The scheme has the benefit of being easy to implement but performs poorly in the presence of high-frequency waves. High order accurate FDTD methods may be derived but ultimately rely on neighboring grid points when approximating derivative.

In contrast to FDTD methods, the Hermite methods of Goodrich and co-authors (2006) use Hermite interpolation and a staggered (dual) grid to construct high order accurate numerical methods for first order hyperbolic equations. These methods achieve high order approximations in both time and space by reconstructing local

polynomials within cells of the computational domain and employing Hermite-Taylor time stepping. The resulting schemes are able to evolve the solution locally within a cell making them ideal for parallel computing. Building on the original Hermite methods this thesis focuses on two goals: (1) the development of new Hermite methods and (2) their implementation on modern computing architectures.

To accomplish the first objective, this thesis presents two variations of Hermite methods which are designed to simplify the scheme while preserving the favorable features. The first variation is a family of Hermite methods which do not require a dual grid. These methods eliminate the need for storing dual coefficients while maintaining optimal convergence rates. The second type of variation are Hermite methods which use leapfrog time-stepping. These schemes propagate the solution with less computation than the original scheme and may be used for either first or second order equations.

To address the second objective, this thesis presents algorithms which take advantage of the many-core architecture of graphics processing units (GPU). As three-dimensional simulations can easily exceed the memory of a single GPU, techniques for partitioning the data across multiple GPUs are presented. Finally, this thesis presents numerical results and performance studies which confirm the accuracy and efficiency of the proposed Hermite methods for linear and nonlinear wave equations.

Acknowledgments

My time at Rice has been one of the most fruitful experiences of my life. I am very grateful and feel fortunate to have been mentored by Dr. Tim Warburton. I owe my success to his approach in mentoring and great attention to detail. In addition he provided the best environment to carry out my graduate studies. I thank Dr. Jesse Chan for serving as a co-mentor and the constant feedback he provided which vastly improved my writing, communication skills, and understanding of the development of mathematics. I thank Dr. Adrianna Gillman for serving as a co-adviser, her feedback regarding public speaking has been truly invaluable. It is the belief that an effective scientist should be able to communicate complex ideas to their peers that has resonated with me the most. I credit this trio of mentors for my success. I would like to thank my committee members Dr. Matt Knepley, and Dr. Stephen Bradshaw for their advice and valuable feedback.

In addition I was fortunate enough to collaborate with Dr. Thomas Hagstrom and Dr. Daniel Appelö, leading experts on Hermite methods. Conversations with them have been crucial to my thesis work and has paved the way for future collaborations.

Dr. Béatrice Riviére once asked me why I wanted to attend Rice University and why graduate school. I explained that to me graduate school would serve as a stepping stone to my dream of working at a national lab. As such graduate school has been exactly that. I credit Rice for the valuable skills which led to various internships in industry and a national lab. I would like to thank Nick Tanushev and Sean Hardesty at Z-Terra, Scott Morton and Thomas Cullison at Hess, and Tzanio Kolev at Lawrence

Livermore National Lab for the opportunity to spend my summers working under their direction.

In addition I would like thank the Rice CAAM department for making me feel so welcomed and at home. I thank my office mate Zheng Wang for listening to me explain my ideas. Finally, I would like to thank Veronica Landa for making these last five years the best.

I acknowledge and thank the National Science Foundation for providing funding to carry out my graduate studies. NSF GRFP grant number: DGE-1450681.

Contents

Illustrations

Tables

Chapter 1

Introduction

The simulation of wave dominated physical phenomena often relies on models based on hyperbolic partial differential equations (PDEs). Much research has been devoted to the development of numerical algorithms for these equations, yet they can still be challenging to solve. For example, when simulating propagation of short waves over many wavelengths, a large number of points per wavelength are needed to accurately approximate the phase and amplitude of the propagating wave. The simulation of propagating waves using low order methods is typically subject to significant numerical dissipation and dispersion. High order methods have the advantage of both rapid convergence and decreased numerical dissipation [4, 5] when compared to low order methods for sufficiently smooth solutions [6, 7]. The downside of high order methods is that they often have a high number of operations per data access, but this is becoming less important on modern computing architectures as computing devices tend have a higher peak flop performance compared to memory bandwidth. This gap in bandwidth favors algorithms with higher flop per data movement [8].

Wave simulation is essential to many fields of study. For example, in geophysics the numerical solution to the acoustic wave equation is central to various imaging algorithms such as Reverse Time Migration [9] and Full Wave Form Inversion [10]. In the context of electromagnetism, electrical engineers employ numerical simulations of Maxwell's equation to aid in the design of new products such as radars and antennae [11]. More complicated wave-like phenomenon may be modeled though nonlinear

equations such as Euler's equations for gas dynamics [12] and the shallow water equations which have applications in weather prediction and tsunami modeling [13, 14].

1.1 Hermite Methods

For the simulation of wave propagation, this thesis focuses on the development of methods based on Hermite interpolation; in particular the methods introduced by Goodrich and co-authors in [15]. Hermite methods are a class of numerical schemes which employ a local polynomial based solution at each cell and maintain a 2^d-stencil in d-dimensions regardless of order. A remarkable feature of these methods is their ability to propagate the solution with a time step determined only by cell-size and speed of wave propagation. In addition, evolution is strictly localized to each cell. This feature is advantageous for high order or multi-stage time stepping methods since it eliminates the need to communicate between neighboring cells during the evolution stage. This feature makes Hermite methods well-suited for parallel implementations [16, 17].

Hermite methods were first proposed for the numerical solution of hyperbolic problems with smooth solutions but have since been applied to equations with discontinuous initial conditions where they have achieved favorable results [18]. As modeling wave propagation in complex geometries is cumbersome with Hermite methods, an effort has been made to hybridize with methods which offer more geometric flexibility [19, 17]. Adaptivity for the methods have been introduced by means of p-adaptivity [20], and preliminary work on h-adaptivity is presented in [17]. The dissipation and dispersion properties of Hermite methods have been demonstrated to be competitive in terms of cost with other high order structured grid schemes [18, 21]. Finally,

although relatively new, Hermite methods have enjoyed success in applications of aero-acoustics [21], electromagnetism [19], and fluid dynamics [22, 23].

1.2 Overview of Numerical Methods

As Hermite methods correspond to a relatively young class of numerical methods it necessitates a comparison over commonly used numerical methods.

1.2.1 Finite Difference Methods

In applications of geophysics and electromagnetism, the finite difference time domain scheme (FDTD) is the workhorse for simulating wave propagation [24, 25]. Finite difference schemes approximate spatial derivatives by difference formulas. The difference formulas are based on dimension by dimension polynomial approximations and rely on structured grids. These schemes are among the simplest and easiest to implement and are well understood from a theoretical perspective [26, 27]. The appealing factors of finite difference schemes are their ease of implementation and fully explicit formulations. In addition, higher order schemes may be constructed by varying the width and/or shape of the associated stencil, the consequence of this is the loss of localized approximations and order independent time-step [28]. FDTD schemes are also ideal for fine grain parallelism and have been successfully mapped to various parallel programming paradigms and specialized hardware such as the graphics processing unit (GPU) [24, 29]. A major drawback of finite difference schemes is the difficulty of simulating waves in complex geometries.

1.2.2 Element based Methods

It is the tensor based designed of Hermite and finite difference methods that make them ill-suited for solving problems on complex geometries. Element based methods introduce geometric flexibility by assuming a general domain may be represented by a collection of elements. These elements are typically quadrilaterals or triangles (hexahedra or tetrahedra in three-dimensions) typically organized in an unstructured manner to fill the physical domain [4, 30, 31]. Finite element methods achieve high order accuracy by representing the solution over each element as a high order polynomial. In addition to the geometric flexibility these methods result in better approximations of gradients and reduced dispersion and diffusion errors when compared to finite difference methods. Finite element methods correspond to a broad family of schemes based on a variational formulation [32], the most well known is the Bubnov-Galerkin scheme [33]. For time-dependent problems, these methods have the burden of a global mass matrix inversion. Techniques such as mass lumping aim to approximate the global mass matrix with a diagonal matrix but reduce the accuracy of the method.

A variation of these methods, discontinuous Galerkin (DG), offer the benefit of geometric flexibility and explicit formulations [4, 34]. These methods have demonstrated to perform exceptionally well for simulating wave propagation. A key feature of these methods is their ability to represent the solution using polynomials which are discontinuous at element interfaces. This ability has utility for modeling wave propagation in media with sharp interfaces. Discontinuous Galerkin methods however, have a higher computational cost compared to the classic finite element scheme on account of their added degrees of freedom. Furthermore for linear wave problems,

DG methods have a time stepping restriction governed by the following estimate:

$$\Delta t \leq C_{CFL} \frac{h}{c(N+1)^2},$$

where c denotes the speed of wave propagation, N corresponds to the order of the polynomial, and h corresponds to the smallest length of an element. The CFL constant, C_{CFL} depends on the integrator used [35]. By comparison, Hermite methods have the following time step restriction:

$$\Delta t \leq \frac{h}{c},$$

and thus can take a much larger time step than DG methods. To improve time-stepping for DG methods, Warburton and Hagstrom have proposed the use of a covolume filter [28], these methods are heavily influenced by the Hermite methods of Goodrich and co-authors [15]. Additional formulations with Bernstein-Bezier basis functions have been proposed to reduce the computational cost of DG at high orders [36]. Despite their drawbacks, discontinuous Galerkin methods have the advantage of being well suited for modern computer architectures [37].

1.2.3 Spectral Methods

The previously discussed methods aim to approximate the solution of a PDE using a collection of local approximations. Spectral methods take a different approach in which the solution is approximated using globally supported smooth basis functions [38, 39, 40]. The choice in basis functions leads to different types of spectral methods. Examples of commonly used basis functions are the trigonometric functions, Chebyshev polynomials, and Legendre polynomials. The strengths in spectral methods lie

in their ability to provide the so-called spectral accuracy, i.e. their rate of convergence is exponential with respect to the order of approximation. Additionally, since spectral methods do not approximate spatial derivatives with difference formulas, spatial derivatives may be approximated exactly for waves supported in the discrete space and thus phase errors are only introduced by time-stepping. A major drawback of these methods is the associated computational cost. For example, the basis functions for a spectral method with trigonometric basis functions will result in dense linear algebra. The computational burden, however, may be reduced by the use of the fast Fourier transform. Unfortunately, it has been the complexity associated with parallel implementations as well as the difficulties in modeling wave propagation in complex geometries that has prevent them from being the method of choice for wave propagation.

1.2.4 A Context for Hermite Methods

The combination of highly localized evolution, an order independent stability criteria, and high order approximations in both time and space make Hermite methods fairly attractive for scientific computing. Similar to finite difference schemes these methods have a discretization which relies on structured grids but have the advantage of being able to maintain highly localized evolution. In contrast to discontinuous Galerkin methods, their time step is independent of the method order. Lastly, in comparison to spectral methods, the operations are purely local making them ideal for parallel programming.

1.3 Objective of Thesis

The objective of this thesis is to contribute to the development of numerical methods based on Hermite interpolation. In developing new methods I seek to simplify the Hermite methods of Goodrich and co-authors and propose new time-stepping strategies which allow for evolution of both first and second order equations. In addition, a major focus of my work is the development of algorithms which take advantage of modern computing architectures. Toward this goal, I dedicate Chapter 5 in which I describe tailoring Hermite methods to graphics processing units.

As a starting point, the following chapter presents an in-depth literature. I begin by discussing theoretical properties of Hermite methods and recent developments. As simplifications of the methods, I present Hermite methods which do not require a dual grid. These dual grid free Hermite methods are demonstrated to be numerically stable while achieving the same rate of convergence as the classic Hermite method. Furthermore, the coupling of Hermite methods and DG is revisited and is demonstrated to be simplified with the new methods. Second, to reduce the cost associated with time-stepping, I present Hermite methods which use leapfrog time-stepping. These new methods require less computation than the standard Hermite-Taylor time stepping scheme and can be used for equations in first and second order form.

To enable large scale simulations, I discuss how to exploit the tensor product structure of the methods on a graphics processing unit (GPU) to accelerate computations. Algorithms are proposed for single and multi-GPU implementations. A detailed analysis of the performance is given along with a description of the optimization techniques used to improve performance. Finally, a variety of examples of simulating wave propagation is presented to illustrate the accuracy and efficiency of the methods.

1.4 Publications and Presentations

At the time of writing the work in this thesis led to contributions in various articles, invited talks, and poster presentations.

1.4.1 Papers

- **A. Vargas**, J. Chan, T. Hagstrom, T. Warburton, Variations on Hermite Methods for Wave Propagation, Accepted to Communications in Computational Physics, 2016. This work introduces Hermite methods which do not require a dual grid and forms the foundation of Chapter 3.

- **A. Vargas**, J. Chan, T. Hagstrom, T. Warburton, GPU Acceleration of Hermite Methods for the Simulation of Wave Propagation, Accepted to International Conference on Spectral and High Order Methods proceedings 2016. This work discusses the implementation and optimization of Hermite methods for a three-dimensional advection equation. The work provides the foundation of Chapter 5.

- D. Appelo, T. Hagstrom, **A. Vargas**, Globally Super-Convergent Dissipative and Conservative Hermite Methods for the Scalar Wave Equation, To be submitted. This article introduces two Hermite methods for the scalar wave equation, a dissipative and a Hermite-Leapfrog scheme for the acoustic wave equation in second order form. The Hermite-Leapfrog scheme was discovered concurrently by the co-authors and me and is described in Chapter 4.

1.4.2 Notable Speaking Events

- Sandia National Lab, Simulating wave propagation with Hermite Interpolation, Albuquerque, New Mexico, January 2017.

- International Conference on Spectral and High Order Methods 2016, Rio de Janeiro, Brazil, A.Vargas, J. Chan, T. Warburton, GPU Accelerated Hermite Methods for the Simulation of Waves.

- Rice Oil and Gas Workshop 2016, Houston, Texas, A.Vargas, J. Chan, T. Warburton, GPU Accelerated Hermite Methods for the Simulation of Waves.

Chapter 2

Hermite Methods

Hermite methods, as introduced by Goodrich and co-authors [15], are a class of numerical methods which represent the solution of hyperbolic PDEs using a piecewise polynomial basis by collocating the solution and its derivatives at the nodes of a structured grid. Hermite methods follow a two-stage procedure in which polynomials (Hermite interpolants) are reconstructed by interpolating the function value and derivatives from the vertices of cells. Evolution is carried out by approximating the function value and derivatives at the midpoint of the cell for the following time step. The second stage of the scheme repeats the processes going from the dual to the primary grid. Figure 2.1 illustrates the Hermite method algorithm.

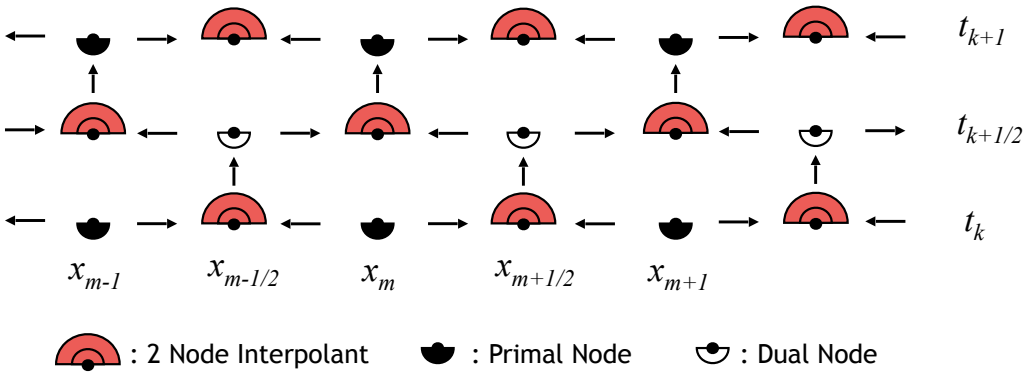


Figure 2.1 : Illustration of the Hermite method algorithm. The Hermite methods of Goodrich and co-authors combine Hermite interpolation and a staggered (dual) grid to produce high order numerical methods. The interpolants approximate the function value and derivatives at the midpoint of a cell. The solution is then evolved on the nodes of the staggered grid. The schemes alternate between interpolating the function value and derivatives of the primary (black) and dual grid (white).

These methods have been demonstrated to be stable and high order accurate for hyperbolic equations, including equations with nonlinearities [15, 17, 22]. The use of Hermite interpolation for the numerical solution of PDEs is not unique to the work of Goodrich et al. Jet Schemes [41, 42, 43] are a generalization of semi-Lagrangian methods for advection equations, which achieve high order approximations by tracking derivative and function data along characteristic curves. Similar to Hermite methods, Hermite interpolants are used to construct polynomials to represent the solution along each cell. The work of Mathioudakis et al. also uses Hermite interpolation to propose a collocation finite element method for solving the Helmholtz equation [44, 45, 46].

2.1 Hermite Interpolation

At the heart of Hermite methods is the Hermite interpolation problem as described by Dahquist and Bjork [47]:

Theorem 2.1.1. *Let $\{x_i\}_{i=1}^{s}$ be s distinct, real, or complex points. Let $f(x)$ be a given real or complex valued function defined with derivatives up to order m_i at x_i. The* ***Hermite interpolation problem****, to find a polynomial $g(x)$ of degree $\leq r-1$ where $r = \sum_{i=1}^{s}(m_i+1)$ such that*

$$\left.\frac{d^j g(x)}{dx_j}\right|_{x=x_i} = \left.\frac{d^j f(x)}{dx_j}\right|_{x=x_i}, \quad j = 0, ..., m_i, \quad i = 1, ..., s, \tag{2.1}$$

is uniquely solvable.

In order to aid in the introduction of Hermite methods, I employ the advection

equation

$$\begin{aligned} \frac{\partial u}{\partial t} &= a\frac{\partial u}{\partial x} \\ u\left(x, t_0\right) &= f(x), \quad x \in [x_{min}, x_{max}) \\ u\left(t, x_{min}\right) &= u\left(t, x_{max}\right), \quad t > 0. \end{aligned} \tag{2.2}$$

Here a denotes a constant wave speed, and $f(x)$ is a smooth initial condition. The spatial discretization is carried out over a primary grid, Ω, defined as a collection of K equispaced points

$$\Omega = \{x_m : \quad x_m = x_{min} + mh_x, \quad m = 0, \ldots, K-1\},$$

where K is the number of grid points on the interval (x_{min}, x_{max}) and $h_x = (x_{max} - x_{min})/K$ denotes the spacing between the nodes. For periodic domains, it is assumed that $x_{m+K} = x_m$. The solution represents a collection of piecewise polynomials each centered at a node, x_m, on the primary grid of the form

$$p_m(x, t_k) = \sum_{l=0}^{N} c_l \left(\frac{x - x_m}{h}\right)^l,$$

at time step $t_k = t_0 + \Delta t k$, where Δt denotes the step size and k denotes the number of steps taken. The scalar h denotes the length of the interpolation interval, in this case, $h = h_x$. By construction, the coefficients of the polynomial are simply the scaled function value and spatial derivatives at a fixed time step t_k

$$c_l = \frac{h^l}{l!}\frac{d^l u(x, t_k)}{dx^l}\bigg|_{x_m}, \quad l = 0, \ldots, N.$$

Hermite interpolation is used to construct a $2N+1$ polynomial to represent the solution over a cell. To facilitate the interpolation process a dual grid, $\tilde{\Omega}$, is introduced

$$\tilde{\Omega} = \left\{ x_{m+1/2} = x_{min} + (m+1/2)\, h_x, \quad j = 0, \ldots, K-1 \right\}.$$

The degree $2N+1$ polynomial over (x_m, x_{m+1}), which is referred to as $p_{m+\frac{1}{2}}(x, t_k)$, is constructed by interpolating the $(N+1)$ scaled function values and spatial derivatives at the left and right endpoints of the interval (x_m, x_{m+1}). The polynomial $p_{m+\frac{1}{2}}(x, t_k)$ is represented using the following expansion

$$Rp_{m+\frac{1}{2}}(x, t_k) = \sum_{l=0}^{2N+1} c_l \left(\frac{x - x_{m+\frac{1}{2}}}{h} \right)^l,$$

where c_l are the function and scaled derivatives at the dual grid node $x_{m+\frac{1}{2}}$. These coefficients are determined by solving the Hermite interpolation problem (Theorem 2.1.1) enforcing the conditions

$$\left. \frac{\partial^i p_{m+\frac{1}{2}}}{\partial x^i} \right|_{x_m} = \left. \frac{\partial^i p_m}{\partial x^i} \right|_{x_m}, \qquad \left. \frac{\partial^i p_{m+\frac{1}{2}}}{\partial x^i} \right|_{x_{m+1}} = \left. \frac{\partial^i p_{m+1}}{\partial x^i} \right|_{x_{m+1}}, \qquad i = 0, \ldots, N.$$

Determining the coefficients for the interpolant may be achieved by solving the linear system

$$\begin{bmatrix} \mathbf{C}^m \\ \mathbf{C}^{m+1} \end{bmatrix} \mathbf{p}_{m+1/2} = \begin{bmatrix} \mathbf{p}_m \\ \mathbf{p}_{m+1} \end{bmatrix}, \tag{2.3}$$

where the constraint matrices, $\mathbf{C}^m, \mathbf{C}^{m+1} \in \mathbb{R}^{(N+1)\times(2N+2)}$, enforce conditions on $p_{m+\frac{1}{2}}$

and its derivatives at the points x_m, x_{m+1},

$$\mathbf{C}_{ij}^{m} = \begin{cases} \left(\frac{x_m - x_{m+\frac{1}{2}}}{h}\right)^{j-i} \frac{1}{i!} \prod_{s=0}^{i-1}(j-s), & j \geq i \\ 0, & j < i \end{cases}$$

$$\mathbf{C}_{ij}^{m+1} = \begin{cases} \left(\frac{x_{m+1} - x_{m+\frac{1}{2}}}{h}\right)^{j-i} \frac{1}{i!} \prod_{s=0}^{i-1}(j-s), & j \geq i \\ 0, & j < i \end{cases}.$$

and $\mathbf{p}_m, \mathbf{p}_{m+1}$ denotes the vector of function and scaled derivatives at each of the nodes. For convenience, $\mathbf{H}$ will denote the interpolation matrix which maps function and scaled derivative data at the endpoints of the interval (x_m, x_{m+1}) to a degree $2N+1$ expansion at the node $x_{m+\frac{1}{2}}$

$$\mathbf{H} = \begin{bmatrix} \mathbf{C}^{j} \\ \mathbf{C}^{j+1} \end{bmatrix}^{-1}.$$

The construction of the type of polynomial may also be carried out through a spline representation [48, 49].

2.1.1 Properties of the Interpolants

Vital to the stability and convergence of Hermite methods are properties of the interpolation operators. While $\mathbf{H}$ corresponded to the interpolation operator acting on discretized function and derivative values, I will introduce the operators $\mathcal{I}$ and $\tilde{\mathcal{I}}$, to correspond to the interpolation operators acting on a continuous level, interpolating from a primary to a dual grid and a dual grid to the primary grid respectively. The following theorems and lemmas employed by Goodrich et al. in [15] are essential to

proving convergence and will play a role in the new methods introduced in Chapter 3 and 4. Following the notation in [15], the L_2-inner product and norm for functions of 2π periodic functions with q weak derivatives is expressed as

$$(f,g) = \int_{-\pi}^{\pi} f(x)\, g(x)\, dx, \quad \|g\|^2 = (g,g),$$

and the Sobolev norm, $\|f\|_{H^q}^2$, which acts on H^q,

$$H^q = \left\{ f : \left(\int_{-\pi}^{\pi} \sum_{i=0}^{q} |D^i f|^2 \right)^{\frac{1}{2}} < \infty \right\},$$

the space of 2π-periodic functions with q weak derivatives in L_2, is expressed as

$$\|f\|_{H^q}^2 = \sum_{v=0}^{q} \|D^v f\|^2.$$

The first lemma which is proved in [50] provides error bound for the interpolation operators.

Lemma 2.1.2 (Birkoff et al. [50])**.** *The Hermite interpolation operators, $\mathcal{I}$ and $\tilde{\mathcal{I}}$, satisfy*

$$\|g - \mathcal{I}g\| \leq Ch^{2N+2}\|D^{2N+2}g\| \quad for \quad g \in H_{per}^{2N+2}, \tag{2.4}$$

$$\|D^{N+1}(g - \mathcal{I}g)\| \leq Ch^{N+1}\|D^{2N+2}g\| \quad for \quad g \in H_{per}^{2N+2}, \tag{2.5}$$

$$\|g - \mathcal{I}g\| \leq Ch^{N+1}\|D^{N+1}g\| \quad for \quad g \in H_{per}^{N+1}. \tag{2.6}$$

A corollary may then be derived by setting $g = q - \mathcal{I}q$ in the last equation and observing $\mathcal{I}g = 0$ for this instance of g.

Corollary 2.1.2.1 (Goodrich et al. [15])**.**

$$\|q - \mathcal{I}q\| \quad \leq \quad Ch^{N+1}\|D^{N+1}(q - \mathcal{I}q)\ \| \quad for \quad q \in H_{per}^{N+1}.$$

As noted by Goodrich et al. in [15] a direct convergence proof in L_2 fails because the L_2 norm of the interpolation operators are not bounded by $1 + O(h)$. Given that the operators are not bounded in an L_2 sense, proof of convergence for Hermite methods relies on the following **Orthogonality Lemma**.

Lemma 2.1.3 (Goodrich et al. [15])**.** *For all f,g* $\in \left(H_{per}^{N+1}\right)$

$$\left(D^{N+1}\mathcal{I}f, D^{N+1}\left(g - \mathcal{I}g\right)\right) = 0.$$

Furthermore, by defining the following semi-inner product

$$(p, q)_{N+1} = \left(D^{N+1}p, D^{N+1}q\right), \quad p, q \in H_{per}^{N+1}$$

and through the use of the Orthogonality Lemma (2.1.3), it can be shown that the interpolation operator $\mathcal{I}f$ is semi-orthogonal to any interpolation error $g - \mathcal{I}g$. Applying the orthogonality lemma (2.1.3) with $g = f$ gives

$$\|D^{N+1}f\|^2 = \|D^{N+1}\mathcal{I}f\|^2 + \|D^{N+1}\left(f - \mathcal{I}f\right)\|^2, \quad f \in H_{per}^{N+1}. \tag{2.7}$$

In particular,

$$\|D^{N+1}\mathcal{I}f\| \leq \|D^{N+1}f\|, \quad f \in H_{per}^{N+1}. \tag{2.8}$$

Statement (2.8) shows that the interpolation process is a contraction on the function data with respect to the semi-norm, $\|\cdot\|_{N+1}$. The stability of Hermite methods hinges

on this contractive property [15].

2.2 Hermite-Taylor Methods

Having described the interpolation procedure and essential properties, the focus now turns to the evolution stage. The structure of the Hermite interpolant allows for an evolution scheme based on a temporal Taylor series expansion and the Cauchy-Kowalevski recurrence relation [51]. Recalling that by means of Hermite interpolation the solution of the PDE is represented at a given time, t_k, by the $2N+1$ polynomial

$$Rp_m(x,t_k) = \sum_{l=0}^{2N+1} c_l \left(\frac{x-x_m}{h}\right)^l,$$

at a cell centered at node x_m. Evolution through the Hermite-Taylor scheme is carried out by constructing a temporal series centered around $t = t_k$, as given by the polynomial

$$Rp_m(x,t) = \sum_{l=0}^{2N+1} \sum_{s=0}^{2N+1-l} c_{l,s} \left(\frac{x-x_m}{h}\right)^l \left(\frac{t-t_k}{\Delta t}\right)^s. \tag{2.9}$$

The task of evolution then reduces to determining the coefficients, $c_{l,s}$, for the proposed space-time polynomial (Equation 2.9). By construction all coefficients are known for $s = 0$ (provided by Hermite interpolation), thus the coefficients for $s > 0$ remain to be determined. Using the model PDE (Equation 2.2) it can be observed that temporal derivatives can be exchanged for spatial derivatives

$$\frac{\partial u}{\partial t} = a\frac{\partial u}{\partial x}.$$

Furthermore, by repeated differentiation, a recurrence relation (Cauchy-Kowelasky) can be observed for sufficiently regular functions:

$$\frac{\partial^s \partial^l}{\partial t^s \partial x^l} = a \frac{\partial^{s-1} \partial^{l+1}}{\partial t^{s-1} \partial x^{l+1}} \quad l = 0, ..., 2N + 1 - s, \quad s = 0, ..., 2N. \tag{2.10}$$

By substituting the space-time polynomial (Equation 2.9) into the recurrence relation (Equation 2.10) and evaluating at a smooth point (x_m, t_k), the PDE can be used to express the remaining unknown coefficients in terms of known coefficients

$$c_{l,s+1} = a \frac{(l+1)\Delta t}{(s+1)h} c_{l+1,s} \quad l = 0, ..., 2N - s, \quad s = 0, ..., 2N. \tag{2.11}$$

Once the coefficients are known, the Hermite space-time polynomial (Equation 2.9) is evaluated at the following half-step leading to the update formula for each degree of freedom,

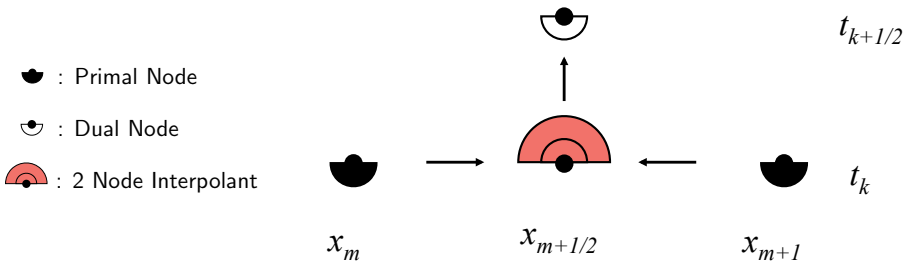
$$\frac{h^l}{l!} \frac{\partial^l}{\partial x^l} u(x_{m+\frac{1}{2}}, t_{k+\frac{1}{2}}) \approx \frac{h^l}{l!} \frac{d^l}{dx^l} p^k_{m+\frac{1}{2}}(x_{m+\frac{1}{2}}, t_{k+\frac{1}{2}}) = \sum_{s=0}^{2N+1} c_{l,s} \left(\frac{1}{2}\right)^s, \quad l = 0, ..., N.$$

Figure 2.2 illustrates the reconstruction and evolution procedure.

2.2.1 Stability and Convergence

In order to show properties of numerical stability and convergence, the model problem is generalized to system

$$\mathbf{u}_t = A\mathbf{u}_x, \quad A = A^T \in \mathbb{R}^{d \times d}, \quad \mathbf{u} \in \mathbb{R}^d \tag{2.12}$$
$$\mathbf{u}(x, t_0) = f(x), \quad f(x + 2\pi) = f(x).$$



(a) Goodrich et al.'s Hermite Methods

Figure 2.2 : Illustration of local reconstruction and evolution. Hermite methods construct a $2N+1$ polynomial (red hat) by interpolating function values and the first N derivatives of subsequent nodes (black half circles). The interpolant is then used to approximate the function values and derivatives at the next time step at the nodes of the dual grid.

As before spatially periodic solutions are assumed and evolution is taken from a fixed time, t_0, to a final time, T. Existing theorems and lemmas on stability and convergence are established by treating the evolution provided by the Hermite-Taylor method to be exact. The first lemma discusses conditions in which the Hermite evolution is stable.

Lemma 2.2.1 (Stability Lemma [15])**.** *Suppose $p(x,t_k)$ is an $\mathbb{R}^d$-valued degree $2N+1$ piecewise polynomial function of x with breaks at $x_0 < x_1 < ... < x_{K-1}$. If $p(x,t)$ satisfies Equation (2.12) for $t \geq t_k$ and the time step size, Δt, satisfies the CFL condition*

$$\rho(A)\,\Delta t < min\left(x_{m+1} - x_{m+\frac{1}{2}}, x_{m+\frac{1}{2}} - x_m\right),$$

then $p(x_{m+\frac{1}{2}}, t_{m+\frac{1}{2}})$ is given by the space-time Taylor series and the coefficients satisfy the vector recurrence relation.

Building from the stability lemma, a proof of convergence is presented by Goodrich et al. in [15] by considering a Taylor series representation of the analytic solution.

Theorem 2.2.2 (Hermite Convergence Theorem [15]). *Let* $\frac{\Delta t}{h} > 0$ *and* $T > 0$ *be fixed. Suppose* $f \in (H^{2N+2}_{per})^d$ *and the hypothesis of Lemma 2.2.1 holds, then there exist, a constant, C independent of h so that*

$$\|u^k - p^k\| + \|u^{k+\frac{1}{2}} - p^{k+\frac{1}{2}}\| \leq Ch^{2N+1}\|D^{2N+2}f\| \quad for \quad 0 \leq k\Delta t \leq T.$$

Theorem 2.2.2 denotes $u^k = u(\cdot, k\Delta t)$ as the solution at the primal grid at time t_k and $u^{k+\frac{1}{2}} = u(\cdot, \Delta t(k+\frac{1}{2}))$ as the step on the dual grid at time $t_{k+1/2}$.

2.2.2 Arbitrary dimension

Hermite methods discretize higher dimensional PDEs by considering dimension by dimension polynomial reconstructions. To aid in the description I introduce the d-dimensional system

$$\frac{\partial u}{\partial t} + \sum_{j=1}^{d} a_j \frac{\partial u}{\partial x_j} = 0, \quad a \in \mathbb{R}. \tag{2.13}$$

In higher dimensions, the computational grid is generated through a tensor product construction. An order N Hermite method requires $(N+1)^d$ degrees of freedom over each grid point and may be stored in the tensor

$$p^i_{m_1,\ldots,m_d}(t_n) \approx \frac{h^{|\alpha|}}{i!} D^i u\left(x_{1,m_1}, \ldots, x_{d,m_d}, t_n\right), \tag{2.14}$$

where h denotes the spacing between the nodes, D denotes the derivative operator, and $i = (i_1, \ldots, i_d)$ denotes the multi-index with i_j ranging from 0 to N. High dimensional Hermite interpolation constructs polynomials of the form

$$Rp_{m_1+\frac{1}{2},\ldots,m_d+\frac{1}{2}} = \sum_{j_1=0}^{2N+1} \cdots \sum_{j_d=0}^{2N+1} b_{j_1,\ldots,j_d} \left(\frac{x_1 - x_{m_1+\frac{1}{2}}}{h_{x_1}}\right) \ldots \left(\frac{x_1 - x_{m_1+\frac{1}{2}}}{h_{x_1}}\right). \tag{2.15}$$

As in the one-dimensional case, determining the function value and partial derivatives at the next time step may be accomplished through a temporal Taylor series and the recurrence relation derived from the PDE.

2.3 Extensions of Hermite methods

Since the introduction of Hermite methods, variations which aim to improve efficiency and address limitations have been proposed in the literature. For example, Hermite-Taylor time-stepping can become computationally expensive in the presence of nonlinearities motivating the need for alternative time-stepping, thus a more efficient Hermite-Runge Kutta scheme is presented in [22]. To address modeling in complex geometries Hermite methods have been coupled with discontinuous Galerkin [19]. Adaptivity for Hermite methods as been introduced by means of p-adaptivity in [20]. To enable large scale computations, Hermite methods have been implemented in parallel programming paradigms for various applications. The described extensions will be the focus of the remaining chapter.

2.3.1 Evolution through Integration

In an effort to combat the expense of Hermite-Taylor evolution, Hagstrom and Appelö have proposed method of lines methodology to evolve the interpolant in [22]. Rather than constructing a space-time polynomial, the $2N+1$ degree polynomial are considered with time-dependent coefficients

$$p_m(x,t) = \sum_{l=0}^{2N+1} c_l(t) \left(\frac{x - x_m}{h} \right)^l . \tag{2.16}$$

The polynomial may then be substituted into the PDE (Equation 2.2)

$$\frac{dp_m}{dt} = a\frac{dp_m}{dx}. \tag{2.17}$$

Evaluating the polynomial at the node where it is centered, x_m, results in the following ordinary differential equation (ODE)

$$\frac{dc_0(t)}{dt} = a\frac{c_1(t)}{h}. \tag{2.18}$$

Repeatedly differentiating Equation 2.17 and evaluating at the midpoint of the cell yields a system of ODEs for the coefficients of the polynomial

$$\frac{dc_l(t)}{dt} = a\frac{(l+1)c_{l+1}(t)}{h^l} \quad l = 0, ..., 2N. \tag{2.19}$$

Numerical studies have demonstrated that so long as the evolution is carried out with sufficient accuracy, the scheme may remain to be stable [22].

2.3.2 Hybrid Hermite Methods

The need to simulate on complex geometries has led to the hybridization of Hermite and discontinuous Galerkin (DG) methods yielding a class of high-order and geometrically flexible solvers. As DG is an arbitrary order finite element scheme, it is able to naturally accommodate complex geometries. DG methods, however, suffer from a time step restriction which depends on order. In comparison, Hermite methods may evolve the solution at a much bigger time step.

The geometric flexibility of DG and the excellent time-stepping abilities of Hermite led to Chen and Appelö's development on hybrid Hermite-DG schemes [19]. Since

Hermite and DG evolve the solution at different rates and require communication of boundary data an appropriate strategy was needed. Chen and Appelö employed semi-structured grids; in which the interior of the geometry is a cartesian grid defining the nodes for the Hermite grid, and the border of the geometry is made up of triangular elements. The elements bordering the Hermite nodes consisted of overlapping Hermite nodes and mesh vertices. Movement of data from Hermite to DG was accomplished by extrapolating from the Hermite interpolant, while movement of data from DG to Hermite was accomplished by fitting a Hermite polynomial, in the l_2 sense, over the DG solution defined on a pair of elements.

2.3.3 P-adaptive Methods

For equations that exhibit highly localized features or sharp transitions, it is often desirable to employ an adaptive approach that allows for higher order approximations where needed. Adaptive schemes have the ability to provide refined approximations thereby minimizing error in regions prone to high error, e.g. singularities and shocks. There are two main approaches to adaptivity: p-adaptivity and h-adaptivity. P-adaptive methods improve approximations by increasing the order of the polynomial in regions of the domain. By contrast, h-adaptive methods improve accuracy in regions of the domain by grid refinement.

Chen and Hagstrom pioneered p-adaptive Hermite methods in [20]. By leveraging the local evolution, these p-adaptive methods construct polynomials of variable order depending on cell location. The selected order of the polynomial is determined by considering the rate of decay of the coefficients. Post evolution the scheme is truncated based on a specified truncation criteria. Preliminary work on h-adaptivity was presented in [17]; however no discussion on how refinement should be carried out was

presented.

2.3.4 Hermite Methods for Conservation Laws

For a general scalar conservation law in one space dimension

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0, \tag{2.20}$$

where u is the state variable and $f(u)$ is the flux (or rate of flow) it can be shown that the Hermite methods of Goodrich and co-authors do not conserve variable quantities. The reason for this is that Hermite methods evaluate fluxes at cell centers and as a consequence do not enforce continuity at the cell interfaces. To address this issue the work of Kornelous and Appelö introduced the flux conservative Hermite methods for the conservation laws [52, 53, 54]. Flux-conservative Hermite methods resolve discontinuous fluxes by computing numerical fluxes at cell edges and interpolating the result to the cell center. It is precisely the added interpolation step that allows for continuous fluxes, though vital to the scheme it does introduce additional interpolation steps. The interpolation step in the flux-conservative Hermite methods is precisely what enables stability. Furthermore, these methods are able to maintain the optimal converge rates $O(h^{2N+1})$ of the classic Hermite method for smooth solutions.

To address shocks, Kornelous and Appelö adopted the work by J.L. Guermond in which an indicator function is used to detect shocks and introduce viscosity as needed [55]. The challenge in accommodating shocks in numerical simulations is the spurious solutions that may be observed when using high-order numerical methods. Typically schemes either introduce a slope limiter [56] or a viscosity introduced near shocks [57].

2.3.5 Efficient Implementations and High Performance Computing

The favorable features of Hermite methods, such as localized evolution and order independent stencils, have made them attractive for multi/many-core architectures. Several studies have been performed examining and suggesting techniques to improve the computational performance.

In the case of nonlinear problems, the operations associated with the Hermite-Taylor scheme can quickly dominate the computational cost. As an alternative using ODE integration techniques can be less expensive [22]. Furthermore, the cost of polynomial multiplication associated with nonlinear PDEs can be reduced by using the fast Fourier transform (FFT) which turns polynomial multiplication to point-wise multiplication.

Further acceleration can be made possible by parallel implementations. Appelö and co-authors present a parallel solver for the compressible Navier-Stokes equation [17]. The parallelization is carried out by the Message Passing Interface (MPI) library. The study demonstrated that Hermite methods can maintain high efficiency and near linear speed ups ranging from 32-512 cores.

With the development of NVIDIA's Compute Unified Device Architecture (CUDA), Graphic Processing Units (GPUs) have become an attractive option for fine grain parallelization. In a preliminary study, Dye implemented a Hermite solver for the two-dimensional advection equation with constant coefficients on a single GPU [16]. This study employed a cell per thread strategy, in which each thread was responsible for performing interpolation and evolution of the Hermite interpolant. Chapter 5 will introduce an alternative strategy which exposes fine-grained parallelism.

2.3.6 Initial Data and Forcing Functions

A difficulty that arises in employing Hermite methods is the requirement of derivative data at nodal values from initial conditions. In higher dimensions, this means supplying mixed derivatives. In an effort to allow for various input functions Appelö and co-authors have employed the univariate Chebychev polynomial software, Chebyfun, to express functions in terms of the Chebychev series [58] in [17]. A Chebychev series takes the form

$$f(x) = \sum_{i=0}^{\infty} c_i T_i(x), \tag{2.21}$$

where $T_i(x)$ is the Chebychev function. Higher dimensional functions are approximated by taking tensor products of the Chebychev functions. Multivariate functions which may be expressed as products of single variable functions are indeed ideal for Hermite methods since they allow for simpler computation of partial derivatives.

2.4 Dispersion and Dissipation Properties

In designing high-resolution numerical methods for wave propagation one seeks numerical methods that minimize dissipation and dispersion errors. An analysis of the dissipation and dispersion properties for Hermite methods has been pioneered by Jang and Appelö in [18, 21, 59]. Their results demonstrated that Hermite methods are more dissipative than dispersive. The analysis is carried out through two approaches. The first approach is through a modified advection equation which provides the leading order of the dissipation error

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = (-1)^N \nu_v \frac{\partial^{2N+2} u}{\partial x^{2N+2}}, \tag{2.22}$$

with

$$\nu_v = \frac{1}{\gamma}\frac{h^{2N+1}}{(2N+2)!}\int_{-1}^{1}\left(\frac{1-(z-\gamma)^2}{4}\right)^{N+1} dz, \tag{2.23}$$

where $\gamma = \frac{\Delta t}{h}$. This approach assumes exact evolution.

The second approach considers a discrete modal analysis. This approach equates the evolution of discrete data over a half-step as multiplication by a block circulant matrix, $A(\gamma, N)$. Numerical experiments demonstrated that Hermite methods achieved improved dissipation error with fewer points per wavelength compared to central difference finite difference schemes. In Chapter 3, I discuss techniques to minimize dispersive and dissipative errors.

2.5 Summary

This chapter provided an in-depth introduction to the Hermite methods of Goodrich et al. along with recent developments. Hermite methods are a class of numerical schemes designed for the solution of hyperbolic differential equations. These methods are carried out by a two-step procedure, the first step interpolates the first $(N+1)^d$ (in d-dimensions) function values and derivatives from the vertices of cells to create a $2N+1$ polynomial centered at the midpoint. The second step is the evolution of the solution. The literature has presented two main methods of evolution. The first is through a temporal Taylor series expansion where coefficients for the series are determined by the Cauchy-Kowalevski recurrence relation. For problems with nonlinearities, Hermite-Taylor methods become costly. As an alternative method of evolution, the local polynomial may be evolved through the use of method of lines methodology.

Since the introduction of Hermite methods, mathematicians have extended the methods to incorporate flexibility in geometry and p-adaptivity [20]. Most recently a flux-conservative method was introduced with shock capturing abilities [52]. Hermite methods have been used to solve a variety of problems in acoustics and fluid dynamics.

This literature review allows for an understanding of the properties of Hermite methods to establish the fundamentals needed for the development of new methods. The following chapter will add to the numerical methods based on Hermite interpolation by introducing Hermite methods which do not require a dual grid.

Chapter 3

Hermite Methods on a Single Grid

The Hermite methods of Goodrich et al. (henceforth Dual Hermite methods) iterate between the reconstruction of polynomials and their evolution on staggered grids. As an effort to bypass the need for a staggered grid this chapter introduces three modifications to the polynomial reconstruction procedure in order to maintain evolution on the primary grid. Removing the need for a dual grid offers various advantages; for example in applications with variable coefficients, coefficients are expanded at each grid point as a Taylor series in order to allow for efficient "right-hand side" evaluations (Section 3.5.2). The removal of the dual grid eliminates the need for storing such coefficients. Additionally, it simplifies the implementation of the Hermite and discontinuous Galerkin coupling introduced in [19]. Each proposed method is demonstrated to be numerically stable while maintaining the optimal convergence rate of the classic Hermite scheme.

3.1 Overview of Methods

In this section I introduce three methods which maintain evolution on a single grid, namely the **Central**, **Upwind**, and **Virtual** Hermite methods. The Central Hermite method uses neighboring nodes to reconstruct polynomials on the primary grid. The Upwind Hermite method performs a directional reconstruction by utilizing an adjacent node. The Virtual Hermite scheme projects to and from the dual grid to

	Stage 1		Stage 2	
	Interpolate	Evolve	Interpolate	Evolve
Dual	$\mathbf{u}_m^k,\ \mathbf{u}_{m+1}^k \rightarrow \mathbf{u}_{m+1/2}^k$	$[t_k, t_k + \Delta t)$	$\mathbf{u}_{m-1/2}^{k+1},\ \mathbf{u}_{m+1/2}^k \rightarrow \mathbf{u}_m^k$	$[t_k, t_k + \Delta t)$
Virtual	$\mathbf{u}_m^k,\ \mathbf{u}_{m+1}^k \rightarrow \mathbf{u}_{m+1/2}^k$	$[t_k, t_k)$	$\mathbf{u}_{m-1/2}^{k+1},\ \mathbf{u}_{m+1/2}^k \rightarrow \mathbf{u}_m^k$	$[t_k, t_k + \Delta t)$
Central	$\mathbf{u}_{m-1}^k,\ \mathbf{u}_{m+1}^k \rightarrow \mathbf{u}_m^k$	$[t_k, t_k + \Delta t)$		
Upwind	$\mathbf{u}_{m-1}^k,\ \mathbf{u}_m^k \rightarrow \mathbf{u}_m^k$	$[t_k, t_k + \Delta t)$		

Table 3.1 : Overview of the different stages for the various Hermite algorithms, $\mathbf{u}_m^k$ represents the solution at the k^{th} time-step at the m^{th} grid point.

maintain evolution on the primary grid. The Virtual Hermite method is equivalent to the Dual Hermite method if one takes a step size of zero on the dual grid. Table 3.1 provides an overview of the different methods.

3.2 Central Hermite

The Central Hermite method, performs the polynomial reconstruction on a node of the primary grid, x_m, by interpolating the function and derivative data located on the neighboring primal nodes (x_{m-1} and x_{m+1})

$$\mathbf{Q}_m = \mathbf{H} \begin{bmatrix} \mathbf{u}_{m-1} \\ \\ \mathbf{u}_{m+1} \end{bmatrix}.$$

Here $\mathbf{H}$ refers to the interpolation operator as used by the Dual Hermite method and $\mathbf{Q}_m$ corresponds to the vector of the reconstructed function value and first N derivatives at the midpoint x_m. Reconstructing solely on the primary grid relieves the need for a dual grid. Maintaining the notation introduced in Chapter 2, h_x refers to the spacing between nodes on the primary grid. Under similar domain of dependence

arguments as the Dual Hermite method mentioned in [15], Central Hermite is stable so long as the time step is chosen to satisfy

$$a\Delta t < h_x,$$

where a denotes the maximum speed of the propagating wave. Figure 3.1 illustrates the Central Hermite algorithm.

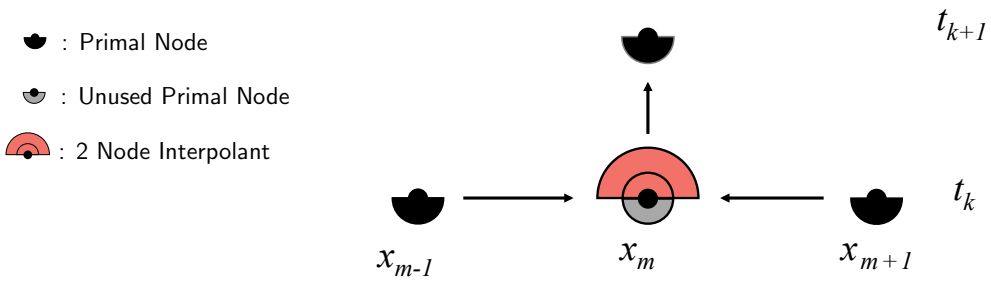


Figure 3.1 : The Central Hermite method constructs an interpolant centered at x_m using nodal information from x_{m+1} and x_{m-1}. The degrees of freedom may then be computed at the following time step for the node x_m via the Hermite-Taylor algorithm or through numerical integration.

3.3 Upwind Hermite

The Upwind Hermite method performs the reconstruction in a directional, or one-sided, manner. The concept of constructing polynomials through one-sided derivatives takes advantage of the directional nature of hyperbolic equations [60, 61] and can be used to enforce boundary conditions [15]. Figure 3.2 illustrates the polynomial construction procedure. The coefficients, $\mathbf{Q}_m$, for the polynomial at node x_m may be constructed by interpolating the function values at the nodes x_{m-1} and x_m.

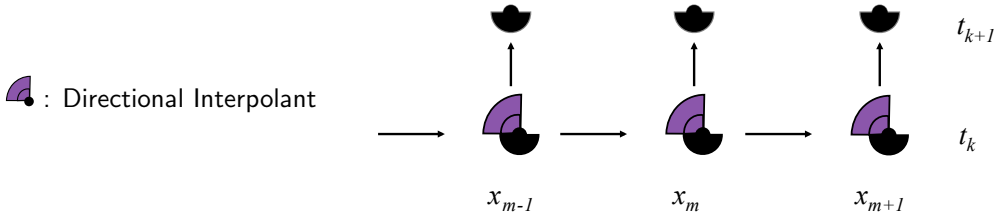


Figure 3.2 : Upwind Hermite interpolates subsequent nodes and centers the polynomial on one of the nodes thereby creating a directional interpolant. Evolution may then be carried out through the Hermite-Taylor algorithm or numerical integration.

Mathematically this is expressed as

$$\mathbf{Q}_m = \mathbf{H}_u \begin{bmatrix} \mathbf{u}_{m-1} \\ \mathbf{u}_m \end{bmatrix}.$$

Here $\mathbf{H}_u$ is an operator unique to the Upwind Hermite scheme. The Upwind Hermite method centers the reconstruction at x_m so the interpolation operator is used to approximate higher order derivatives at x_m. A similar construction is performed when constructing polynomials with nodal data at x_m and x_{m+1}. The Upwind Hermite method maintains the same stability criterion as the Central Hermite method

$$a\Delta t < h_x.$$

In the case of the advection equation (Equation 2.2) the domain of dependence relies on the direction of the propagating wave. More precisely the solution at time $t_k+\Delta t$ at the node, x_m, is only dependent on the solution of $x_m - a\Delta t$. This dependence provides insight on performing the reconstruction, for example when $a > 0$ the polynomial reconstruction at x_m should be performed with the nodes x_{m-1} and x_m in order to

maintain stability. Furthermore, for more complex equations in which directionality is not apparent, a similar procedure may be adapted to reconstruct upwind and downwind characteristics. This approach has been seen in the Weighted Essentially Non-Oscillatory (WENO) reconstructions [62, 63].

3.4 Virtual Hermite

The Virtual Hermite method reconstructs a polynomial by performing a projection to and from the dual grid. This operation may be collapsed into a single linear operator $\mathbf{H}_v$. To reconstruct the function value and derivatives at x_m, $\mathbf{Q_m}$, the operator $\mathbf{H}_v$ acts on the nodal data at nodes x_{m-1}, x_m, and x_{m+1}. Mathematically this is expressed as

$$\mathbf{Q}_m = \mathbf{H}_v \begin{bmatrix} \mathbf{u}_{m-1} \\ \mathbf{u}_m \\ \mathbf{u}_{m+1} \end{bmatrix}.$$

Figure 3.3 provides an illustration of the Virtual Hermite method.

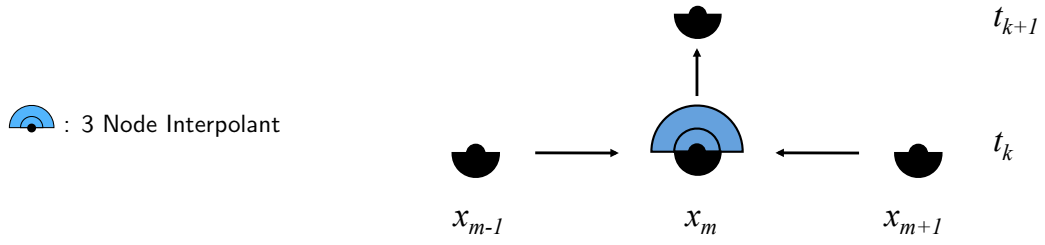


Figure 3.3 : The Virtual Hermite method uses three nodes to perform a projection to and from the dual grid. The resulting polynomial is of degree $2N+1$. Evolution may then be carried out through the Hermite-Taylor algorithm or numerical integration.

The construction of $\mathbf{H}_v$ can be accomplished through the use of the classic Hermite interpolation operator, $\mathbf{H}$, and a truncation operator $\mathbf{R}$. The task of the truncation

operator is to extract the first $(N+1)$ values from the solution vector. Explicitly

$$\mathbf{H}_v = \mathbf{H} \begin{bmatrix} \boxed{\mathbf{RH}} & \mathbf{0} \\ \mathbf{0} & \boxed{\mathbf{RH}} \end{bmatrix},$$

where $\mathbf{0} \in \mathbb{R}^{(N+1)\times(N+1)}$ and $\mathbf{RH} \in \mathbb{R}^{(N+1)\times(2N+2)}$. The matrix $\mathbf{H}_v$ resembles the co-volume filter, as studied in [28], in that it suppresses spurious gradients of the solution on the primal grid. The Virtual Hermite method is able to maintain the same time step restriction as the Dual Hermite method, $a\Delta t < h_x/2$ for the model Equation 2.2.

3.4.1 Proof of Convergence

Under the assumption of Hermite-Taylor evolution, a proof of convergence is provided for the case of constant coefficients.

Theorem 3.4.1 (Virtual Hermite Convergence Theorem)**.** *Let $\frac{\Delta t}{h} > 0$ and let $T > 0$ be fixed. Suppose $f \in (H^{2N+2}_{per})^d$ and that the hypothesis of Lemma 2.2.1 holds. Then there exist a constant C, independent of h so that*

$$\|u^k - p^k\| \leq Ch^{2N+1}\|D^{2N+2}f\| \quad for \quad 0 \leq k\Delta t \leq T.$$

Here u defines the solution to the model system (2.12) and $u^k = u(\cdot, k\Delta t)$ denotes the solution at step k with time step size Δt. The poof of the Theorem is provided in Appendix A.

3.5 Numerical Experiments in One Dimension

Numerical experiments with the Hermite variants are carried out studying convergence rates and qualitative behavior. The experiments were conducted on the bi-unit domain using Hermite-Taylor evolution to solve the advection and acoustic wave equations. Furthermore a CFL constant, $C_{CFL} > 0$, was used as a parameter to modify the time step

$$\Delta t = C_{CFL}\frac{h_r}{a}.$$

Here a is the wave speed and h_r is the radius of the interpolation interval. For the Virtual and Dual methods, h_r corresponds to $h_r = h_x/2$ while the Central and Upwind Hermite methods take $h_r = h_x$. Setting $C_{CFL} \approx 1$ allows for the largest possible time step based on the stability criterion, while setting $C_{CFL} \ll 1$ will result in more time steps than necessary based on the stability criterion.

3.5.1 Convergence Rates

The first set of experiments solved Equation 2.2 with wave speed $a = 1$ and the analytic solution

$$u(x,t) = \sin(\pi(x-t)).$$

The CFL constant is varied between $C_{CFL} = .1$, $C_{CFL} = .5$, and $C_{CFL} = .9$, and solution is propagated to a final time of $T = 10$. Figure 3.4 illustrates the convergence rates and errors for the various Hermite schemes. The error is computed using the usual L_2 norm

$$\|u_a - u_h\|_{L_2} = \sqrt{\int_\Omega (u_a - u_h)^2}.$$

The analytic solution is represented by u_a and u_h corresponds to the approximated solution. Numerical experiments demonstrate that new Hermite methods maintain the optimal $O(h^{2N+1})$ convergence rates. Notably, the Central Hermite method has a higher error due to the larger interpolation interval. The best approximation is achieved by the upwind Hermite method which is to be expected as it is tailored for the advection equation.

	$C_{CFL} = .1$			$C_{CFL} = .5$			$C_{CFL} = .9$		
Order - N	1	2	3	1	2	3	1	2	3
Dual	2.72	4.99	7.02	2.93	5.0	6.98	3.02	5.02	7.01
Virtual	2.67	5.0	7.00	2.96	5.00	7.02	2.99	5.01	7.07
Upwind	2.94	4.99	6.98	2.96	5.0	7.02	3.02	5.03	7.03
Central	1.71	4.94	7.06	2.62	4.98	6.92	2.92	4.98	6.99

Table 3.2 : The L^2 rates of convergence for N^{th} order Dual, Virtual, Upwind, and Central Hermite methods for the one-dimensional advection equation with a smooth sinusoidal solution. The results are presented for various CFL constants, C_{CFL}.

In addition, the growth of the error in time was studied for the Hermite Variants. Error estimates for time-dependent hyperbolic problems have the form

$$e(T) \leq (C_1 + C_2 T) h^{r(N)}$$

where $e\,(T)$ is the error at T, $r(N)$ is the rate of convergence dependent on the polynomial discretization, and C_1 and C_2 are constants [4, 64]. Furthermore a linear growth of error in time for all the Hermite variants (Figure 3.5) was observed with the exception of $C_{CLF} = 1$ which is discussed in the following section.

Errors for the wave equation are also studied through dispersive and dissipation properties of the numerical method [65]. Figure 3.6 illustrates the result of periodic propagation of the function, $e^{-4\sin(\pi x)^2}$, over five periods for local polynomials $N =$

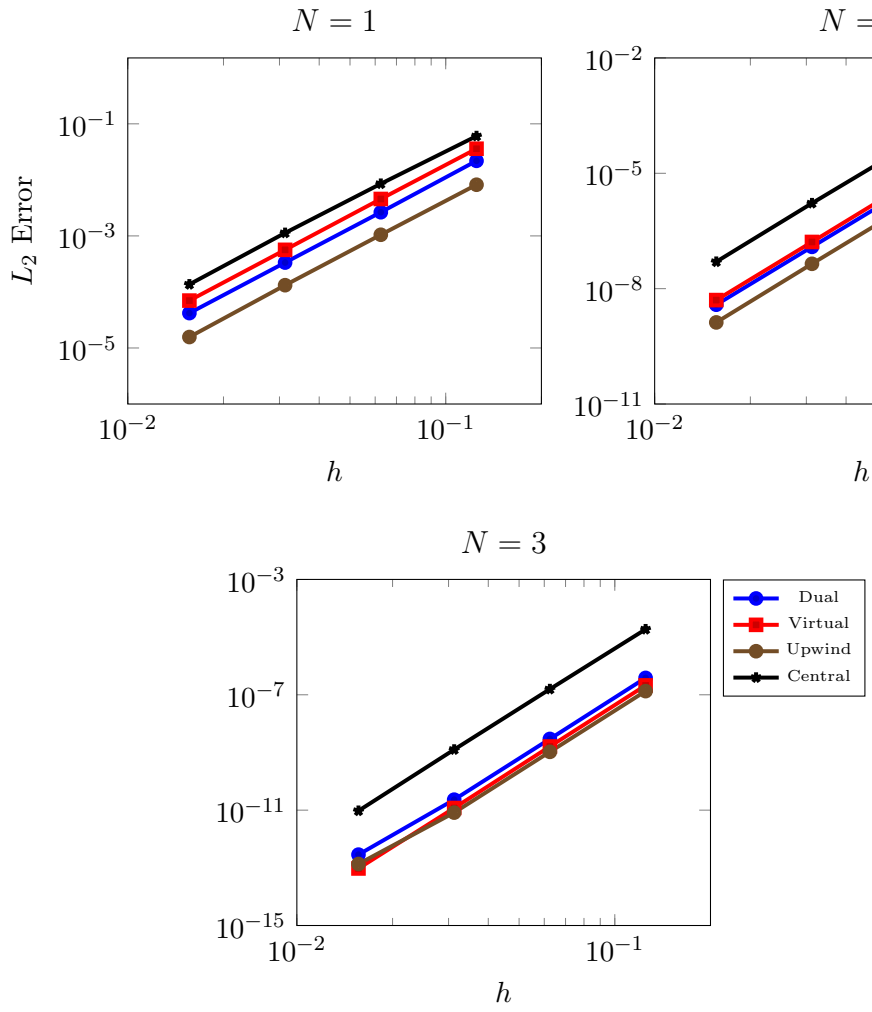


Figure 3.4 : The L^2 errors comparing N^{th} order Dual, Virtual, Upwind, and Central Hermite methods for the one-dimensional advection equation with a smooth sinusoidal solution assuming a CFL constant of $C_{CLF} = 0.9$. For this example the Upwind Hermite method provides the best approximation, this is not surprising as it exploits the directionality of the equation.

.

$1, 2, 3$ and 8 cells. As expected each method resulted in diffusive behavior at low polynomial orders and a low CFL constant, C_{CFL}. Since the Hermite interpolant acts as a smoothing operator, it was observed that for small values of C_{CFL} the solution was overly damped which led to higher errors than with a CFL constant

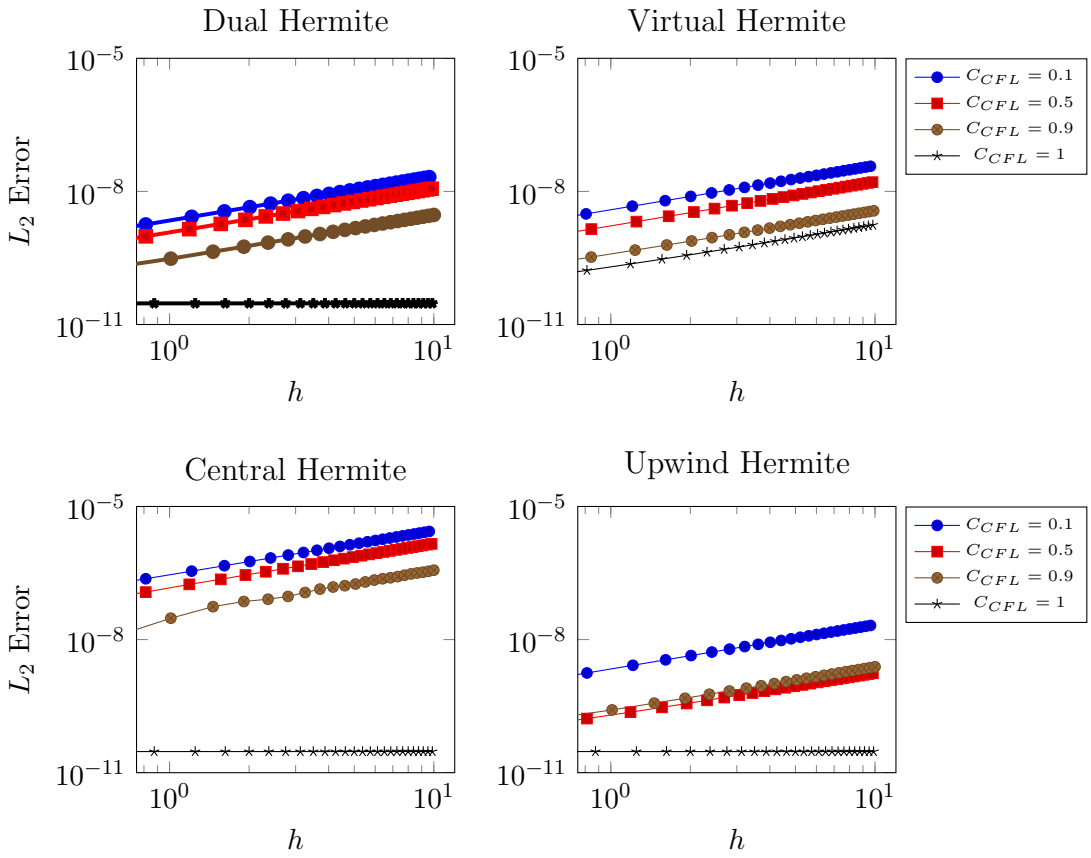
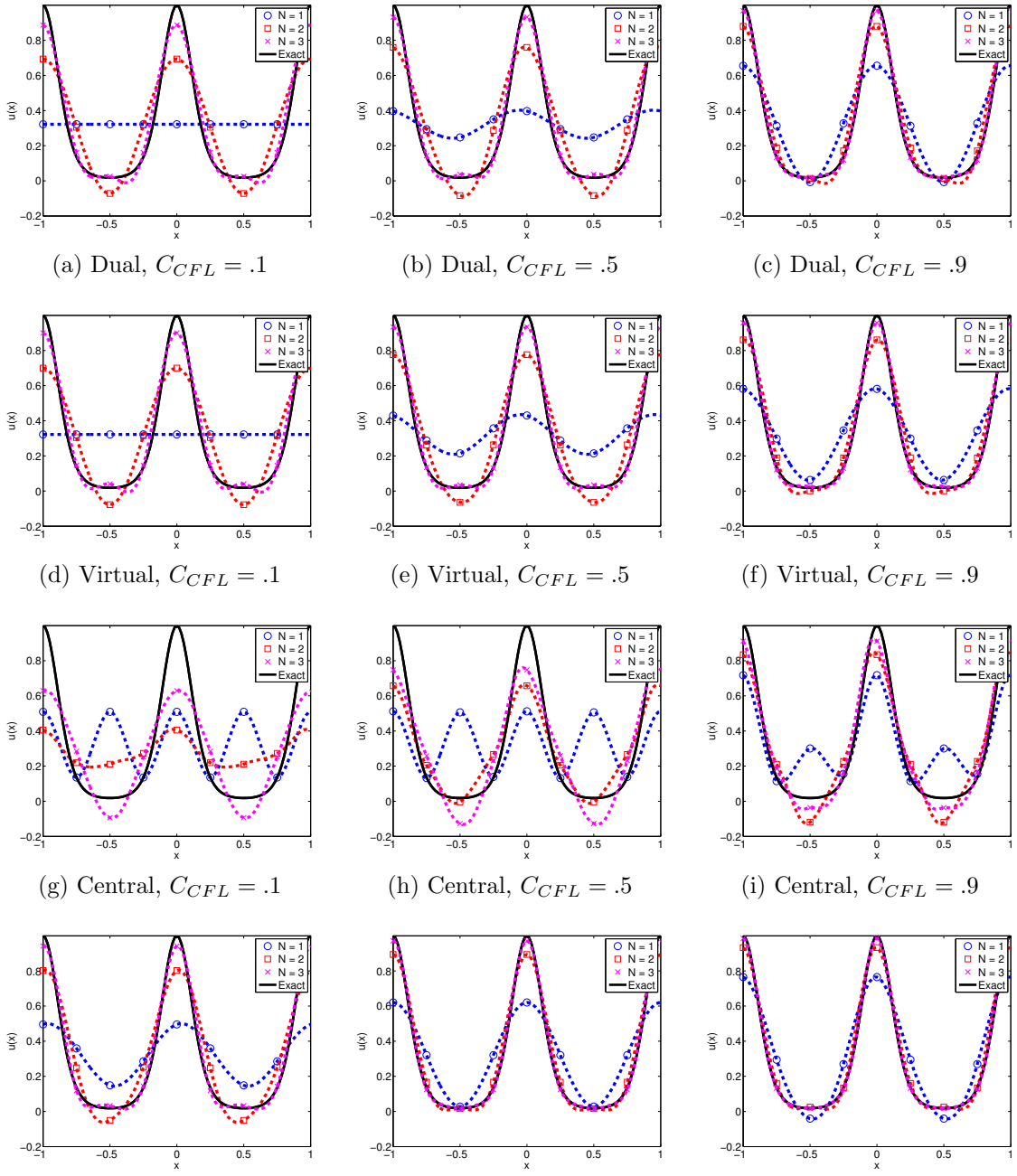


Figure 3.5 : The growth of L^2 error in time for various Hermite schemes using an order $N = 3$ method, $K = 16$ in one dimension. The advection equation is solved up to time $T = 10$ with a smooth sinusoidal solution, and the L^2 error is computed at each time step. A CFL constant, C_{CFL} is introduced to choose the time step.

close to 1. Furthermore, due to the increased interpolation interval, Central Hermite performed poorly compared to the other schemes. Increasing to a finer mesh, $K = 16$, resulted in more comparable results to the Dual Hermite method.

3.5.2 Variable Coefficients and Nonlinearities

For completeness, experiments were carried out with equations with smooth coefficients and nonlinearities.



(a) Dual, $C_{CFL} = .1$ (b) Dual, $C_{CFL} = .5$ (c) Dual, $C_{CFL} = .9$

(d) Virtual, $C_{CFL} = .1$ (e) Virtual, $C_{CFL} = .5$ (f) Virtual, $C_{CFL} = .9$

(g) Central, $C_{CFL} = .1$ (h) Central, $C_{CFL} = .5$ (i) Central, $C_{CFL} = .9$

(j) Upwind, $C_{CFL} = .1$ (k) Upwind, $C_{CFL} = .5$ (l) Upwind, $C_{CFL} = .9$

Figure 3.6 : Advection of a periodic Gaussian pulse by N^{th} order Hermite schemes with various CFL constants on a grid of $K = 8$ cells.

Variable Coefficients

As an example of solving a linear equation with smoothly varying coefficients I consider the following advection equation

$$\frac{\partial u}{\partial t} = a(x)\frac{\partial u}{\partial x} + f(t,x)$$

where spatially varying wave speed is defined by $a(x) = (1 + \sin(x)/2)$. When spatially varying coefficients are introduced the Cauchy Kowelaski recurrence requires the product rule when spatially differentiating the PDE

$$\frac{\partial^n u}{\partial x^n \partial t} = \frac{\partial^n [a(x)\frac{\partial u}{\partial x}]}{\partial x^n} + \frac{\partial^n f}{\partial x^n}.$$

Here n corresponds to the degree of freedom (order of spatial derivative). In the presence of spatially varying coefficients it becomes advantageous to express the coefficients and forcing terms as local Taylor series expansions at each grid point. By expressing in terms of local polynomials, each right hand side evaluation may be computed as polynomial multiplications. As this equation also has a time-dependent forcing function, expanding the function as a space-time polynomial greatly simplifies the evaluations.

For these numerical experiments the forcing term is chosen such that the analytic solution is $u = \cos(16(x+t))$ and the coefficients are expanded around each grid point as order $2N+1$ polynomials. The domain is chosen to be $[0, 2\pi]$ and partitioned into 80, 100, and 120 equally sized cells and the solution is propagated to a final time of $T = 1.25$. Figure 3.7 reports the accuracy of the methods for a fixed CFL constant, $C_{CFL} = 0.9$. Results for lower CFL constants achieve similar convergence rates but

with decreased accuracy, the exception occurs when using a method order $N = 1$ as it exhibits increased dissipation due to the interpolation procedure (Section 3.5.3). Table 3.3 presents the observed rates of convergence for the Hermite methods under varying CFL constants. In contrast to the numerical experiments with constant wave speed, these numerical experiments show that the Dual and Virtual Hermite method provide a better approximation compared to the Upwind Hermite method. Furthermore these numerical experiments demonstrate that the new Hermite methods are able to maintain the standard $O(h^{2N+1})$ convergence rate as the classic Hermite method.

	$C_{CFL} = .1$			$C_{CFL} = .5$			$C_{CFL} = .9$		
Order - N	1	2	3	1	2	3	1	2	3
Dual	2.05	4.93	7.01	2.52	5.00	6.94	2.74	5.00	6.88
Virtual	2.11	4.92	7.00	2.57	5.00	7.01	2.72	4.98	7.07
Central	0.98	4.55	7.05	1.55	4.80	6.90	2.11	4.74	6.70
Upwind	2.58	4.98	6.98	2.76	5.00	6.98	2.81	4.98	7.01

Table 3.3 : The L^2 rates of convergence for N^{th} order Dual, Virtual, Central, and Upwind Hermite methods for the one-dimensional advection equation with variable coefficients and a time-dependent forcing function.

Nonlinear Equations

As a nonlinear example, I use the viscous Burger's equation

$$\frac{\partial u}{\partial t} = -u\frac{\partial u}{\partial x} + \epsilon\frac{\partial^2 u}{\partial x^2}.$$

For nonlinear equations, Hermite-Taylor schemes become computationally expensive when computing time derivatives of nonlinear equations. As the terms are approximated using local Taylor-series expansions, computing a single time derivative re-

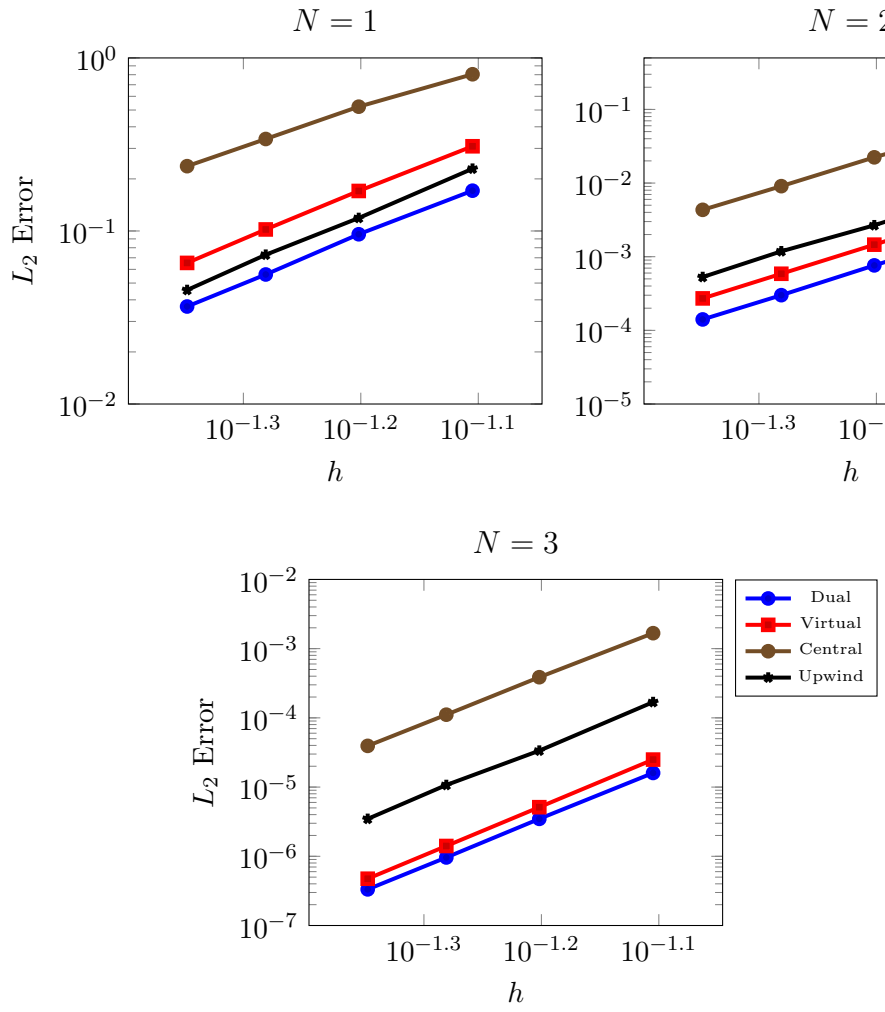


Figure 3.7 : The L^2 rates of convergence for N^{th} order Dual, Virtual, Central, and Upwind Hermite methods for the one-dimensional advection equation with variable coefficients and a time-dependent forcing function. The CFL constant is fixed set to $C_{CFL} = 0.9$.

quires polynomial multiplication between the polynomial approximations of the u and u_x terms. Arbitrary order time derivatives may be computed by the following

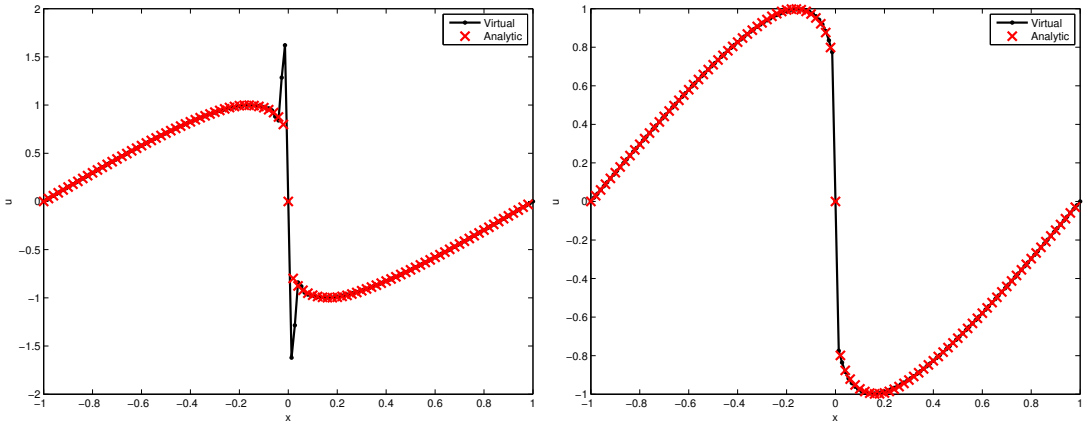
recursion

$$D_t^j u = -\sum_{k=0}^{j-1} D_t^k u D_t^{j-k} u_x + \epsilon D_t^{j-1} u_{xx}. \tag{3.1}$$

Unfortunately for an order N method polynomial multiplication scales as $O(N^2)$ as the j^{th} time derivative is computed. For these numerical experiments, I use a classic fourth order Runge-Kutta scheme where only a single-time derivative needs to be computed [15, 22, 66]. An overview of this approach is presented in Section 2.3.1. Here the initial condition is set to $u(x, 0) = -\sin(\pi x)$ and the computational domain is chosen to be the bi-unit domain.

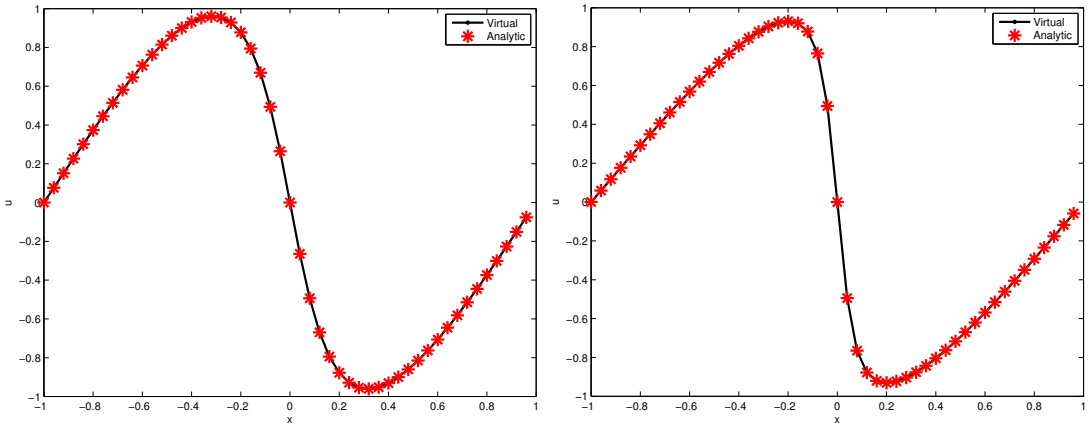
For this example, the solution develops a narrow transition zone at $T = 0.3$, thus the approximation is studied before and after the formation of the transition. To illustrate the resolving power of Hermite methods, Figure 3.8 shows an under resolved and resolved transition at $T = 0.35$ using 50 and 150 grid points for a small epsilon value ($\epsilon = 1e - 3$). For these numerical experiments are carried out using a scheme of order $N = 2$ with a time step chosen by taking $C_{CFL} = 0.1$. Hermite methods with a symmetric stencil were able to adequately resolve the transition. In the presence of diffusion, a purely biased Upwind Hermite method was not stable due to its directional reconstruction.

To study accuracy and convergence, ϵ is set to $\epsilon = 0.02$ and the domain was partitioned in to 25, 35, 45, and 55 equally spaced cells. Table 3.4 reports the accuracy before and after the transition, and Figure 3.9 illustrates the solutions.



(a) Under Resolved Transition (b) Resolved Transition

Figure 3.8 : Approximation of the viscous Burger's equation with an under resolved approximation and a resolved approximation with a viscous term set to $\epsilon = 1e-3$ using an $N = 2$ order Virtual Hermite method.



(a) $T = 0.2$ (b) $T = 0.35$

Figure 3.9 : Approximation of the viscous Burger's equation before (T=0.2) and after the transition (T=3.5) using the Virtual Hermite method with 50 grid points and the viscous term set at ($\epsilon = 0.02$) using an $N = 2$ order Virtual Hermite method.

3.5.3 Spectra and Dispersion/Dissipation Relations

To gain further insight on the new methods, the spectra and dispersion properties were studied. An analysis on the spectra is accomplished by considering a linear

	$T = .2$				$T = .35$			
K - Cells	25	35	45	55	35	45	55	65
Dual	3.3(-5)	6.75(-6)	1.87(-6)	6.24(-7)	1.9(-4)	6.02(-5)	2.21(-5)	8.36(-6)
Rate	-	4.7	5.1	5.5	-	4.68	4.98	5.83
Virtual	6.22(-5)	1.31(-5)	3.78(-6)	1.33(-6)	3.59(-4)	1.16(-4)	4.54(-5)	1.88(-5)
Rate	-	4.6	4.9	5.2	-	4.48	4.69	5.26
Central	7.16(-4)	1.65(-4)	5.26(-5)	2.04(-5)	3.6(-3)	1.3(-3)	5.52(-4)	2.26(-4)
Rate	-	4.4	4.5	4.7	-	4.07	4.26	4.47

Table 3.4 : The L^2 rates of convergence for the second order ($N = 2$) Dual, Virtual, Central, and Upwind Hermite methods for the one-dimensional viscous burgers equation with a smooth sinusoidal solution before ($T = 0.2$) and after the transition ($T = 0.35$).

operator $\mathbf{S}$ which performed the interpolation and evolution procedure for a full time step

$$\mathbf{U}^{k+1} = \mathbf{S}\mathbf{U}^{k}.$$

As the Dual Hermite method considers a full time step propagating the solution from primal to dual and dual to primal, two time steps of the Virtual Hermite method were considered to propagate the solution to the same time

$$\mathbf{U}^{k+2} = \mathbf{S}\mathbf{S}\mathbf{U}^{k}.$$

For further simplification the matrix $\mathbf{A}$ will be defined as the arbitrary update matrix. For the Dual, Central, and Upwind Hermite methods, setting the CFL constant to $C_{CFL} = 1$ resulted in exact evolution. This is unique to constant coefficient equations where the update operator takes the form of a circulant shift matrix and results in $O(h^{2N+2})$ convergence rates, coinciding with [15, Lemma 3.1]. Unfortunately this exactness property was not observed for the Virtual Hermite method.

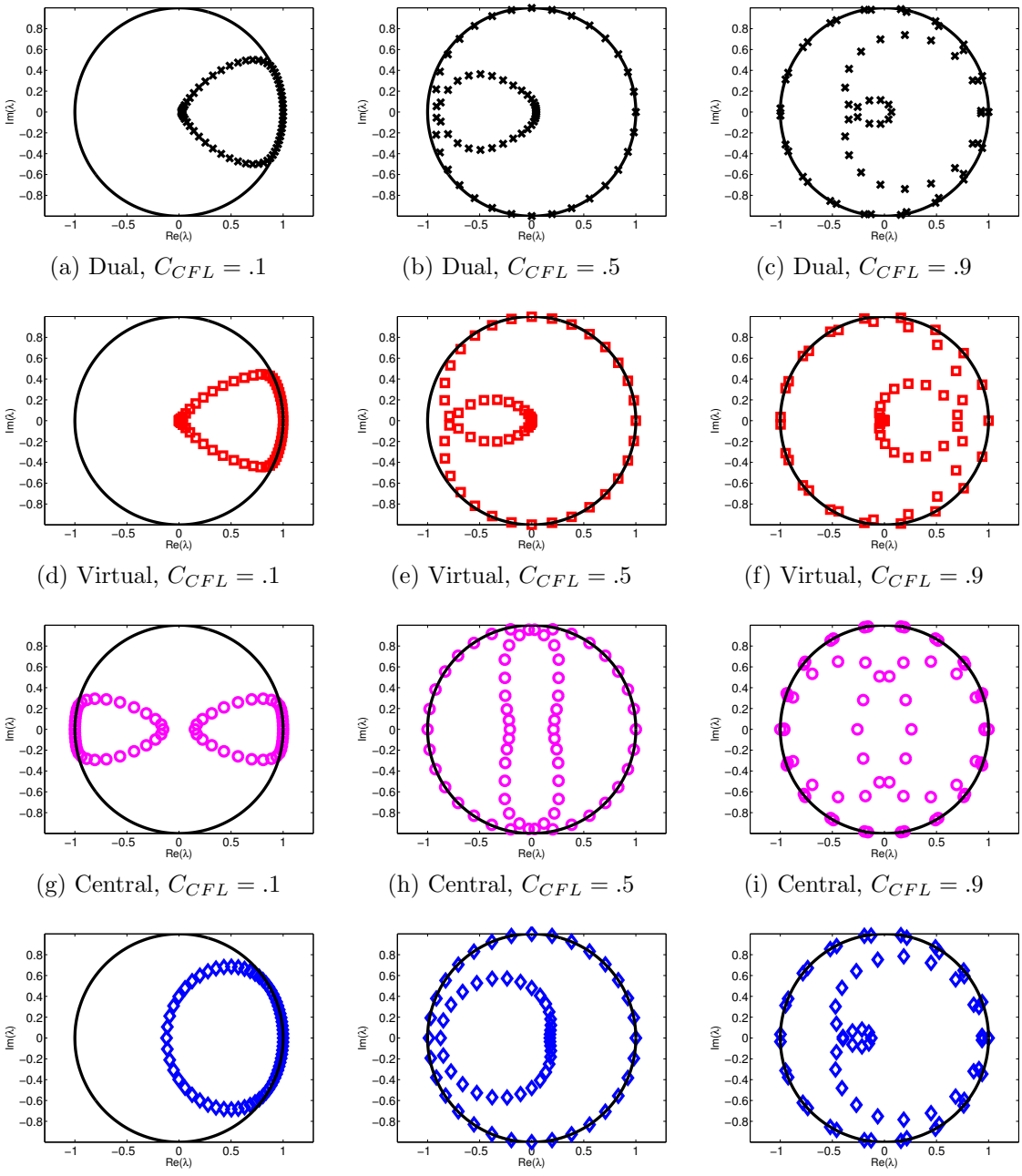
Figure 3.10 illustrates the spectra for the update matrix, $\mathbf{A}$, of the Hermite Vari-

ants. The distribution of the eigenvalues suggest that the Virtual and Dual Hermite methods should behave similarly. Eigenvalues that are found on the unit circle are of the form $e^{i\omega}$ and are related to non-dissipative propagation of modes of the form $e^{i\omega(x-ct)}$. For example, for $C_{CFL} = .1$ the eigenvalues fall closest to the unit circle around the point $(1, 0)$ corresponding to the non-dissipative propagation of modes with small ω (low frequency modes). The remaining spectra lie within the unit circle indicating dissipation of under-resolved modes. The spectra for the Central Hermite method clusters not only around $(1, 0)$ but also around $(-1, 0)$, suggesting that under-resolved high frequency modes may be propagated without dissipation. These spurious modes may explain the behavior of the Central Hermite method observed in Figure 3.6, where propagation of a Gaussian on a coarse grid resulted in "spurious" oscillatory behavior and remained over several periods of advection.

A study on the dissipation and dispersion error for the Dual Hermite method is studied in [21] through a Bloch wave analysis in one dimension. The dissipation and dispersion of the Virtual Hermite method was studied in a similar manner. The analysis was dependent on the Virtual Hermite interpolation matrix, $\mathbf{H}_v$

$$\mathbf{H}_v = [\mathbf{H}_L, \mathbf{H}_C, \mathbf{H}_R]$$

where $\mathbf{H}_C, \mathbf{H}_L$, and $\mathbf{H}_R$ act on Hermite data associated with a given node and its left/right neighbors to produce a reconstruction. For a periodic grid the associated update matrix of size $\mathbf{S} \in \mathbb{R}^{K(2K+2)\times K(2K+2)}$ is a block tridiagonal matrix taking the



(a) Dual, $C_{CFL} = .1$ (b) Dual, $C_{CFL} = .5$ (c) Dual, $C_{CFL} = .9$

(d) Virtual, $C_{CFL} = .1$ (e) Virtual, $C_{CFL} = .5$ (f) Virtual, $C_{CFL} = .9$

(g) Central, $C_{CFL} = .1$ (h) Central, $C_{CFL} = .5$ (i) Central, $C_{CFL} = .9$

(j) Upwind, $C_{CFL} = .1$ (k) Upwind, $C_{CFL} = .5$ (l) Upwind, $C_{CFL} = .9$

Figure 3.10 : Spectra of the update matrix for each Hermite scheme at various CFL constants. The order of order of the method and grid size are fixed to be $N = 3$ and $K = 16$, respectively.

following form

$$\mathbf{S} = \begin{bmatrix} \mathbf{S}_L & \mathbf{S}_R & & \dots & \mathbf{S}_L \\ \mathbf{S}_L & \mathbf{S}_C & \mathbf{S}_R & & \\ & \mathbf{S}_L & \mathbf{S}_C & \ddots & \\ & & \ddots & \ddots & \\ \mathbf{S}_R & & & \mathbf{S}_L & \mathbf{S}_C \end{bmatrix}.$$

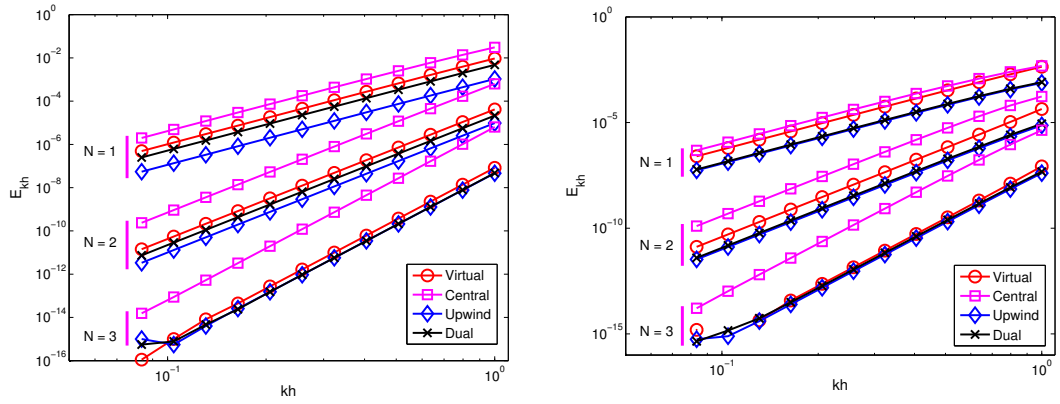
For the Upwind Hermite method $\mathbf{S}_C$, and $\mathbf{S}_R$ are zero (or $\mathbf{S}_L$ depending on the reconstruction); while for the Central Hermite method $\mathbf{S}_C$ is zero. The Bloch analysis was carried out by representing the wave solution $e^{ik(x-ct)}$ in the Hermite basis and noting that the solution was shifted in both space and time by scaling with a complex exponential

$$u(x, t + \Delta t) = u(x, t)e^{-ikc\Delta t}, \qquad u(x + h_x, t) = u(x, t)e^{ikh_x}.$$

Assuming a uniform grid spacing, the discrete evolution of the interpolated exact solution at a node x_m from time t_k to $t_k + \Delta t$ is then given by

$$\left(e^{-ikh_x}\mathbf{S}_L + \mathbf{S}_C + e^{ikh_x}\mathbf{S}_R\right)\mathbf{u}_m^k = \lambda_{h_x}\mathbf{u}_m^{k+1}.$$

Since there are differences in stability criteria between the Hermite Variants, the Dual Hermite was evolved two time steps, propagating from the primary grid to the dual grid back to the primary grid. To match the evolution, the Virtual Hermite method was also evolved two time steps. As the Upwind and Central Hermite method allow for a large time step, a single time step was taken. The dispersion errors, E_{kh_x}, are illustrated in Figure 3.11 over a range of kh_x. The Central Hermite method shows more errors than the Upwind Hermite which shows the smallest errors. Smaller values



(a) $C_{CFL} = .1$ (b) $C_{CFL} = .9$

Figure 3.11 : Dispersive and dissipative errors $\left|\lambda_{h_x} - e^{-ikc\Delta t}\right| / \left|e^{-ikc\Delta t}\right|$ for each Hermite scheme, with order of the method $N = 1, 2, 3$ methods. The computed errors are observed to behave as $O\left(kh_x/c\right)^{2N+2}$.

of the CFL constant C_{CFL} increase the dispersion and dissipation error, though the effect is less noticeable as N increases. For each method the error E_{kh_x} was observed to follow

$$\frac{\left|\lambda_h - e^{-ikc\Delta t}\right|}{\left|e^{-ikc\Delta t}\right|} \approx C_{N+1} \left(\frac{kh_x}{c}\right)^{2N+2}$$

where $N+1$ is the number of degrees of freedom per node and the underlying Hermite approximation space is of degree $2N + 2$.

Optimizing Dispersive and Dissipative Errors

Finally, motivated by Dispersion Relation Preserving (DRP) finite difference schemes [67], dispersive and dissipative errors may be improved through optimization of entries of the interpolation matrix. As mentioned in Section 3.4, the Virtual Hermite method produces a degree $2N+1$ reconstruction using degree N data from three nodes, though

there is sufficient data to define a higher $3N+2$ degree reconstruction. I introduce the interpolation matrix $\tilde{\mathbf{H}} \in \mathbb{R}^{(3N+3)\times(3N+3)}$ given by

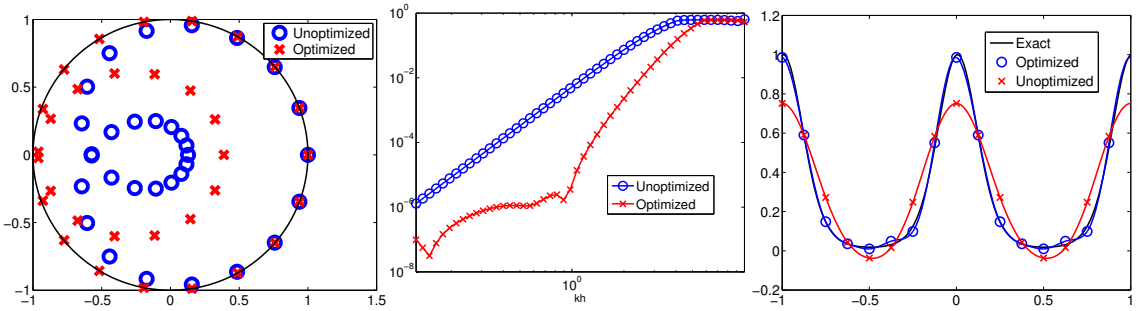
$$\tilde{\mathbf{H}} = \begin{bmatrix} \mathbf{H}_v \\ \mathbf{H}_2 \end{bmatrix}$$

where $\mathbf{H}_v$ is the Virtual Hermite interpolation matrix. DRP schemes enforce a fixed order of approximation for a given finite difference stencil, while using additional degrees of freedom to optimize the dispersion relation. Similarly, by fixing the first $2N+2$ rows of $\tilde{\mathbf{H}}$, the reconstruction implied by $\tilde{\mathbf{H}}$ is enforced to match that of the Virtual Hermite reconstruction for the first $2N+2$ coefficients. The remaining entries of the matrix $\mathbf{H}_2$ (which determine higher order coefficients) are then used to minimize dispersion and dissipation errors. The entries of $\mathbf{H}_v$ depend on the ratio between $L_m - \tilde{x}$ (or $R_m - \tilde{x}$) and as the grid spacing h. Since this ratio is constant as a function of h, $\mathbf{H}_2$ does not change drastically as a grid is refined. However, the optimization does appear to be sensitive to the value of CFL constant.

To demonstrate the effect of optimization, the Virtual Hermite method is compared to an optimized scheme for $N = 1$ and CFL constant $C_{CFL} = .9$. The submatrix $\mathbf{H}_2$ is produced by minimizing the real and imaginary parts of the relative dispersion error for the advection equation

$$\left(\frac{\mathrm{Re}\left(\lambda_h - e^{-ikcdt}\right)}{\mathrm{Re}\left(e^{-ikcdt}\right)} \right)^2 + \left(\frac{\mathrm{Im}\left(\lambda_h - e^{-ikcdt}\right)}{\mathrm{Im}\left(e^{-ikcdt}\right)} \right)^2$$

with unit wave speed and $K = 8$ grid cells. The same optimized submatrix $\mathbf{H}_2$ is then used on a finer $K = 16$ grid, and computed solutions for the Virtual and optimized Hermite methods are compared for the initial condition $e^{-4\sin(\pi x)^2}$ in Figure 3.12. The



(a) Spectra of $\mathbf{S}$ (b) E_{kh} (c) Convection over 5 periods

Figure 3.12 : Spectra, dispersion/dissipation errors, and convection of the initial condition $e^{-4\sin(\pi x)^2}$ over 5 periods. Results are shown for both standard and optimized $N = 1$ order Virtual Hermite methods and $K = 16$ cells, and a CFL constant of $C_{CFL} = .9$. Optimization is done on a coarse $K = 8$ mesh.

spectra of the update matrix $\mathbf{S}$ and dispersion/dissipation errors E_{kh} are also compared in Figure 3.12. The dispersion error and spectra are shown to be significantly improved, and numerical results indicate that under-resolved features are convected with greater accuracy.

Unfortunately, the benefits of such an approach appear to be limited to low orders of approximation. At higher orders, optimization did not reduce the dispersion and dissipation error E_{kh} significantly compared to the unoptimized scheme. Additionally, the stability of such an approach is not guaranteed for Hermite methods (compared to DRP schemes, which optimized over symmetric stencils to guarantee stability). For example, when $N = 1$ the spectral radius of the update matrix for the unoptimized scheme was computed to be $\rho(\mathbf{S}) = 1$ to machine precision. For the $N = 1$ optimized scheme, $\rho(\mathbf{S}) \approx 1.0005$, and strict enforcement of $\rho(\mathbf{S}) \leq 1$ resulted in either non-convergence of the optimization problem or sub par dispersion and dissipation properties.

3.6 Extensions to 2D

The extension of Hermite methods to higher dimensions can be done naturally through the use of tensor products. Assuming grid spacings h_x, h_y in the x and y directions respectively, each point $(x_m, y_m) \in \Omega$, allows for the tensor-product expansion

$$u_m(x,y) = \sum_{j=0}^{2N+1} \sum_{k=0}^{2N+1} \mathbf{u}_{jk} \left(\frac{x - x_m}{h_x} \right)^j \left(\frac{y - y_m}{h_y} \right)^k .$$

The Hermite-Taylor scheme can also be used to evolve the solution. In higher dimensions the evolution procedure retains its simple structure. Extending the Hermite-Taylor series to two dimensions leads to

$$u_m(t,x,y) = \sum_{s=0}^{q} \sum_{j=0}^{2N+1} \sum_{k=0}^{2N+1} \mathbf{u}_{sjk} \left(\frac{t - t_k}{\Delta t} \right)^s \left(\frac{x - x_m}{h_x} \right)^j \left(\frac{y - y_m}{h_y} \right)^k ,$$

where the series truncates exactly for $q = 4(2N + 1)$ [15]. In practice so long the temporal expansion is taken to be $q > 2N + 1$ the optimal rate of convergence may be achieved without having to take a smaller time step. Algorithm 1 illustrates this process for the two-dimensional advection equation with wave speed a

$$u_t = -au_x - au_y,$$

using the operators $\mathbf{D}_x$ and $\mathbf{D}_y$ which correspond to spatial differentiation in x and y, respectively. The symbol $\tilde{\mathbf{u}}$ represents the reconstructed polynomial.

Algorithm 1 Time evolution procedure for 2D scalar advection.

procedure TWO-DIMENSIONAL TEMPORAL TAYLOR SERIES EVALUATION
  $\mathbf{w} = \tilde{\mathbf{u}}^n$
  for $\ell = 4N+2, 4N+1, \ldots, 0$ **do**
    $\mathbf{w} = \mathbf{w} + \frac{\Delta t}{1+\ell}(-a\mathbf{D}_x - a\mathbf{D}_y)\tilde{\mathbf{u}}^n$
  $\tilde{\mathbf{u}}^{N+1} = \mathbf{w}$

3.7 Numerical Experiments in Two Dimensions

Numerical experiments were carried out in two dimensions using the advection and acoustic wave equations. In two dimensions the advection equation is given by

$$\frac{\partial u}{\partial t} + c_x \frac{\partial u}{\partial x} + c_y \frac{\partial u}{\partial x} = 0,$$

where $\mathbf{c} = (c_x, c_y)$ is defined as the unit vector and c_x and c_y are scalars indicating directionality of the wave. The two dimensional wave equation is given by

$$\frac{1}{c^2}\frac{\partial p}{\partial t} = -\nabla \cdot \mathbf{u}, \qquad \frac{\partial \mathbf{u}}{\partial t} = -\nabla p,$$

where c is a specified wave speed, p is pressure, and $\mathbf{u} = (u, v)$ is the velocity. Both equations assume periodic boundary conditions. The stability condition for the two-dimensional advection equation is

$$\|\mathbf{c}\|\Delta t < h_r.$$

For the wave equation the stability condition is simply $c\Delta t < h_r$, where the value h_r is specified by the employed Hermite method. Carrying out similar experiments as performed in one-dimension a CFL constant, C_{CFL}, is introduced as parameter to

determine time step. Furthermore, the Upwind Hermite experiments were restricted to the advection equation due to its explicit directionality. Figure 3.13 reports the errors and convergence rates for the numerical experiments with the advection equation assuming an analytic solution of

$$c_x = \cos(\pi/3), \quad c_y = \sin(\pi/3), \qquad u(x, y, t) = \sin(\pi(x - c_x t)) \sin(\pi(y - c_y t)).$$

In these numerical experiments it can be observed that the Upwind Hermite method outperforms the other Hermite variants. Trailing in performance is the Central Hermite method.These results remain consistent with the one-dimensional numerical experiments.

A similar convergence study was carried out for the two dimensional acoustic wave equation using the exact solution

$$p(x, y, t) = \sin(\pi x) \sin(\pi y) \cos(\sqrt{2}\pi t).$$

Figure 3.14 plots the L^2 errors in p at time $T = 1$ for the Dual, Virtual, and Central Hermite methods. The Dual and Virtual Hermite methods produced errors of very similar magnitude while the error for the Central Hermite method was larger by a factor of roughly 2^N as observed in one dimension. At $C_{CFL} = .9$ and $N = 3$, the error for the Virtual Hermite method was lower than that of the Dual Hermite method. The L^2 rates of convergence are reported in Table 3.5.

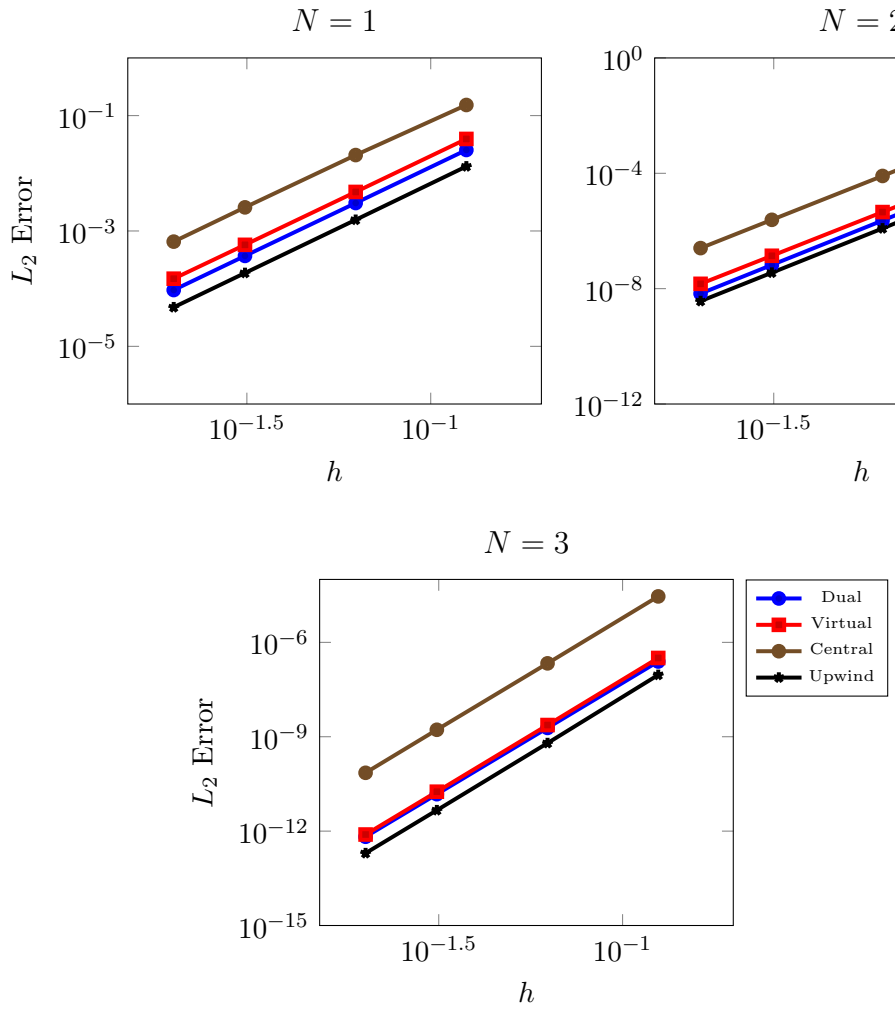


Figure 3.13 : Comparison of L^2 errors for an N^{th} order Hermite scheme for the two-dimensional advection equation with a smooth sinusoidal solution with a CFL constant of $C_{CFL} = 0.9$.

3.8 Coupling with Discontinuous Galerkin

As pioneered by Chen and Appelö in [19], Hermite methods may also be coupled to discontinuous Galerkin methods in order to solve PDEs in more general geometries [19]. Since the Hermite methods introduced in this work do not require staggered grids the transfer of information is simplified. To differentiate between the order of

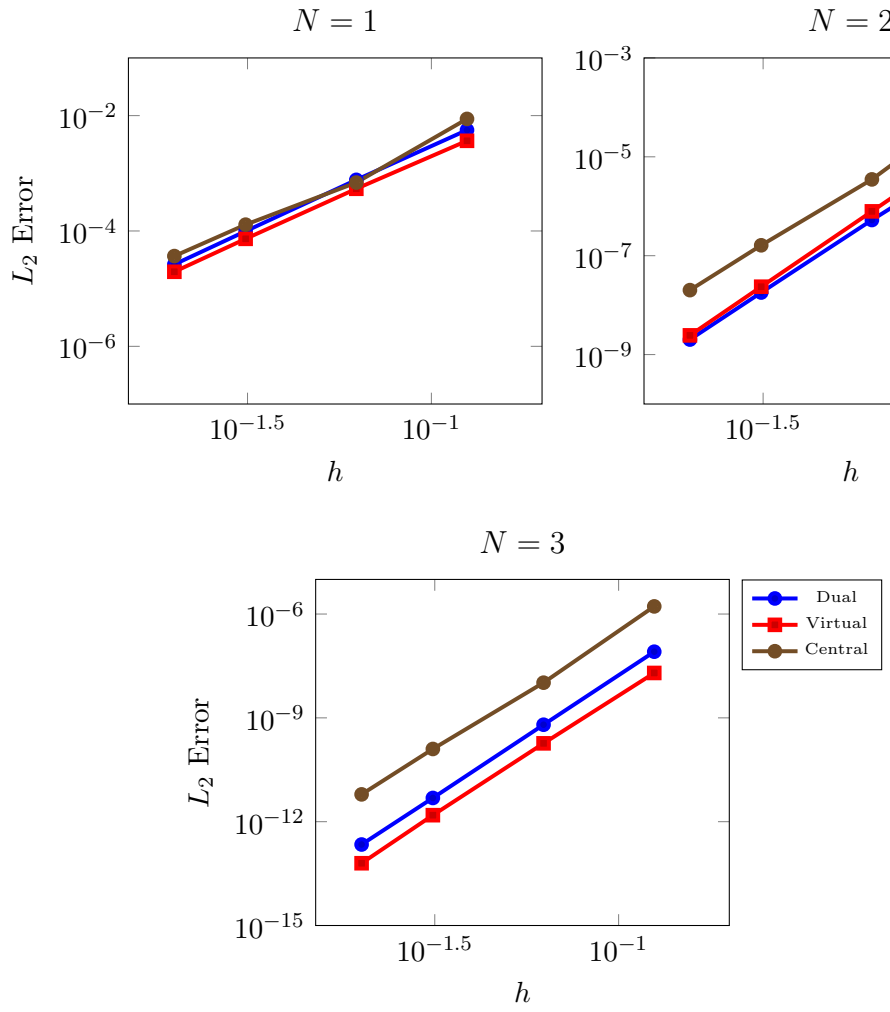


Figure 3.14 : Comparison of the L^2 errors for an N^{th} order Dual, Virtual, and Central Hermite schemes for the acoustic wave equations in two dimensions with $C_{CFL} = 0.9$.

the DG scheme and Hermite, the polynomial order for DG will be referred to as m.

Hermite methods transfer information to DG methods through a numerical flux. Due to a time step restriction of $O(h_x/m^2)$ for DG compared to the $O(h_x)$ time step restriction for Hermite methods, multiple DG substeps must be taken for each Hermite time step. In the numerical experiments, a fourth order Runge-Kutta scheme with five stages was using to evolve the DG scheme and required the evaluation of the

	$C_{CFL} = .1$			$C_{CFL} = .5$			$C_{CFL} = .9$		
N	1	2	3	1	2	3	1	2	3
Dual	2.86	4.93	6.79	2.84	4.73	6.92	2.91	4.77	7.02
Virtual	2.85	4.93	6.84	2.75	4.96	7.03	2.82	5.07	6.83
Central	2.51	4.71	6.05	2.56	4.22	6.82	3.05	4.95	6.84

Table 3.5 : The L^2 rates of convergence of an N^{th} order Dual, Virtual, and Central Hermite methods when applied to the two-dimensional acoustic wave equation with varying CFL constants C_{CFL}.

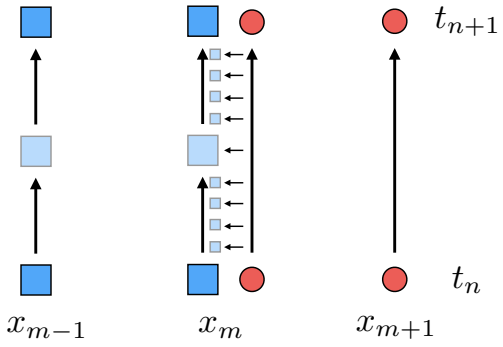


Figure 3.15 : Coupling between Hermite and DG methods without staggered grids (DG nodes are squares, while Hermite nodes are circles). The illustration shows that Hermite methods take a significantly larger time step than DG schemes requiring interpolation between time steps to transfer data from Hermite to DG.

numerical flux for each stage. To maintain high order convergence, flux contributions were computed by evaluating the high order Hermite interpolant in time, as shown in Figure 3.15. The coupling from DG to Hermite is more fragile, as high order derivative information must be determined from the DG solution. In the conducted experiments these derivatives were computed via a patch reconstruction at Hermite nodes by taking derivatives of the reconstructed polynomial [68]. The L^2 errors and convergence rates were computed using the smooth solution

$$p(x, y, t) = \sin(2\pi x)\sin(2\pi y)\cos(2\pi t),$$

and specifying the wave speed to $c^2 = \sqrt{1/2}$. On non-overlapping grids, a polynomial of degree $2N + 1$ was constructed using both Hermite and DG data on neighboring elements. Since the best possible convergence rate for DG is $O(h^{m+1})$,* the order of approximation for DG was taken to be $m = 2N$, where N is the degree of the Hermite method, in order to preserve the $O(h^{2N+1})$ convergence rate.

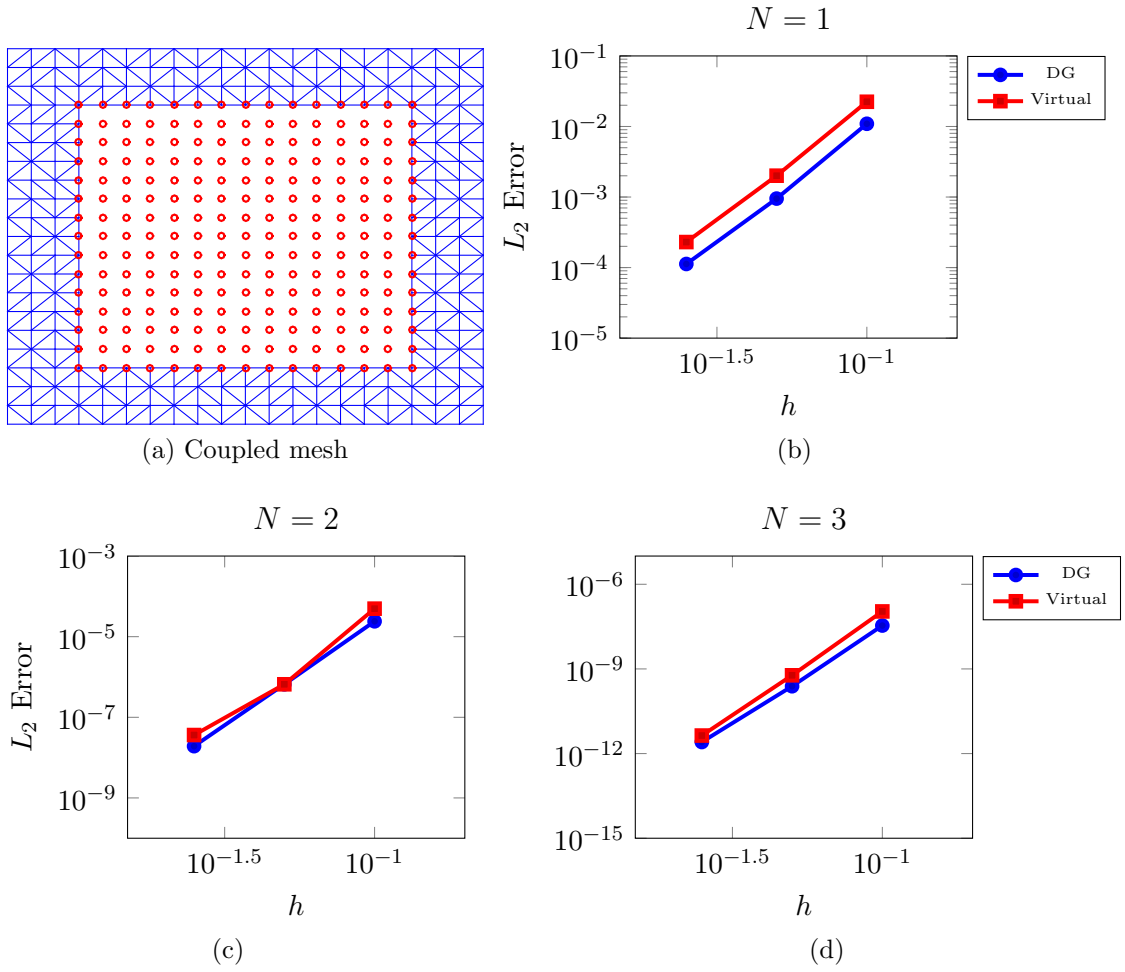
	$N = 1$ / $m = 2$	$N = 2$ / $m = 4$	$N = 3$ / $m = 6$
Virtual Hermite	3.30	5.14	7.30
Discontinuous Galerkin	3.30	5.20	6.85

Table 3.6 : Observed rates of convergence for the coupled Hermite-DG Scheme. In coupling the methods an N^{th} order Virtual Hermite scheme was paired with an m^{th} order DG scheme. This choice allowed the methods to match the their epected rates of convergence, h^{2N+1} and h^{m+1} from the Hermite and DG schemes respectively.

An illustration of the coupled grid used to generate results is found in Figure 3.16 along with the observed accuracy for orders $N = 1, 2, 3$. The coupling in these numerical experiments was carried out using the Virtual Hermite method and the time step was chosen with a CFL constant of $C_{CFL} = 0.8$ for all all orders. For $N > 2$, convergence was limited by the order of the DG Runge-Kutta scheme, and a smaller time step had to be taken to recover optimal convergence rates for the DG solver. Similar observations have been made when coupling Dual Hermite and DG methods [19].

Unfortunately, the approximation of Hermite coefficients using DG becomes less accurate at higher orders, as roughly an order of convergence is lost per derivative.

*Optimal convergence rates for upwind DG are typically observed in practice and are provable on specific classes of meshes [69]. However, on general meshes, DG methods can expect at most $O(h^{m+1/2})$.



(a) Coupled mesh (b) (c) (d)

Figure 3.16 : (a) Example of a coupled mesh for the coupling of the Virtual Hermite and DG methods. The Hermite nodes are denoted by red circles and the triangulation corresponds to the DG discretization. (b-d) Convergence of L^2 errors in 2D for the acoustic wave equations using an N^{th} order Hermite method and $2N$ order DG scheme.

3.9 Summary

This chapter presented the first variation of Hermite methods. In particular, I introduced three variations which do not require dual grids, namely the Central, Upwind, and Virtual Hermite methods. The first method, Central Hermite uses neighboring nodes to reconstruct polynomials on the primary grid. The Upwind Hermite method

exploits the directionality of hyperbolic equations to perform directional reconstructions utilizing adjacent nodes. Finally, the Virtual Hermite methods projects to and from the dual grid to maintain evolution on the primary grid thus the scheme is equivalent to the Dual Hermite method if one takes a step size of zero on the dual grid.

To assess performance and accuracy of these methods, this chapter presented a series of numerical experiments with one and two-dimensional wave equations. These experiments included constant and spatially varying wave speed. In addition, the Central and Virtual Hermite method were applied to non-linear equations. Numerical experiments illustrated that the Upwind Hermite method outperformed the Hermite variants when solving the advection equation with constant wave speed. The Central Hermite method was shown to provide the least accurate approximation on account of the larger interpolation interval. Furthermore, the numerical experiments demonstrated numerical stability and illustrated that these methods can achieve the same rate of convergence as the Dual Hermite method. In the following chapter I introduce a variation of Hermite method which uses leapfrog time stepping.

Chapter 4

Hermite-Leapfrog Methods

This chapter introduces a new family of Hermite methods which use leafrog time-stepping [70, 71, 72, 73]. The new Hermite-Leapfrog methods are high-order accurate multi-step methods which may be used for equations in either first or second order form. Using the acoustic wave equations, as both a pressure-velocity system and a single second order equation, a detailed description of the new methods is presented. The Hermite-Leapfrog scheme for the second order acoustic wave equation was discovered concurrently by Dr. Daniel Appelö and I. This discovery has resulted in the article in progress "Globally Super-Convergent Dissipative and Conservative Hermite Methods for the Scalar Wave Equation." Numerical experiments are presented which demonstrate the stability and efficiency of the new methods.

4.1 The Acoustic Wave Equations

The acoustic wave equations as a pressure-velocity is expressed as

$$\frac{\partial p}{\partial t} + \nabla \cdot \mathbf{v} = 0 \tag{4.1}$$

$$\frac{\partial \mathbf{v}}{\partial t} + \nabla p = 0, \tag{4.2}$$

where $p(\mathbf{x}, t)$ corresponds to the pressure field and $\mathbf{v}(\mathbf{x}, t)$ is the velocity vector. These equations may be simplified to a single second-order PDE by relating the pressure and velocity equations. Precisely, applying a time derivative to the pressure equation

and divergence to the velocity equations yields

$$\frac{\partial^2}{\partial t^2} p + \nabla \cdot \frac{\partial}{\partial t} \mathbf{v} = 0$$
$$\nabla \cdot \frac{\partial}{\partial t} \mathbf{v} + \nabla \cdot \nabla p = 0.$$

Eliminating the velocity term yields the single second order equation

$$\frac{\partial^2 p}{\partial t^2} - \Delta p = 0. \tag{4.3}$$

Both formulations have been widely used in practice and various numerical methods have been proposed for the numerical solution of these equations [74, 75, 76, 77]. In applications such as seismic imaging, these equations form the foundation of imaging algorithms such as "Reverse Time Migration" and "Full Waveform Inversion" [76, 77, 78]. Furthermore, depending on the desired discretization or boundary conditions it may be more convenient to work with one formulation over the other. In this chapter I introduce new methods for both first and second order equations.

4.1.1 Leapfrog Discretization

The so-called "leapfrog" scheme corresponds to a central difference approximation of time derivatives. For example, the first order time derivative is approximated as

$$\frac{\partial p(\mathbf{x}, t)}{\partial t} \approx \frac{p(\mathbf{x}, t + \Delta t) - p(\mathbf{x}, t - \Delta t)}{2\Delta t}, \tag{4.4}$$

and second order time derivative is approximated as

$$\frac{\partial^2 p(\mathbf{x}, t)}{\partial t^2} \approx \frac{p(\mathbf{x}, t + \Delta t) - 2p(\mathbf{x}, t) + p(\mathbf{x}, t - \Delta t)}{\Delta t^2}. \tag{4.5}$$

These discretizations are fairly popular due to their ease of implementation but have the drawback of only being second order accurate. Thus for high order spatial discretizations it may be necessary to take smaller time steps to prevent temporal errors from dominating. The Hermite-Leapfrog schemes presented in this chapter build on leapfrog time-stepping but are high-order accurate methods in both time and space.

4.2 High-Order Leapfrog Time-Stepping

The starting point for the Hermite-Leapfrog schemes is the Taylor series expansion around $t = t + \Delta t/2$ and $t = t - \Delta t/2$. Here I expressed the expansions with respect to the pressure variable p

$$p(\mathbf{x}, t + \Delta t/2) = \sum_{j=0} \frac{(\Delta t/2)^j}{j!} \frac{\partial^j p}{\partial t^j} \tag{4.6a}$$

$$p(\mathbf{x}, t - \Delta t/2) = \sum_{j=0} \frac{(-\Delta t/2)^j}{j!} \frac{\partial^j p}{\partial t^j}. \tag{4.6b}$$

Subtracting Equations 4.6a and 4.6b yields

$$p(\mathbf{x}, t + \Delta t/2) - p(\mathbf{x}, t - \Delta t/2) = \sum_{j=1,odd} \frac{(\Delta t/2)^j}{j!} \frac{\partial^j p}{\partial t^j}, \tag{4.7}$$

which eliminates the even order derivatives yielding an update formula for equations with odd time derivatives. Adding equations 4.6a and 4.6b eliminates the odd order derivatives

$$p(\mathbf{x}, t + \Delta t/2) + p(\mathbf{x}, t - \Delta t/2) = \sum_{j=0,even} \frac{(\Delta t/2)^j}{j!} \frac{\partial^j p}{\partial t^j}, \tag{4.8}$$

yielding an update for equation with even time derivatives.

4.3 Description of the Hermite-Leapfrog Schemes

The temporal Taylor series presented in Section 4.2 forms the foundation of the Hermite-Leapfrog schemes. For clarity, the schemes are first introduced using one-dimensional versions of the pressure-velocity system and the second order acoustic wave equation.

4.3.1 Pressure-Velocity System

In one-dimension the pressure-velocity wave system simplifies to

$$\frac{\partial p}{\partial t} = -\frac{\partial v}{\partial x}, \quad \frac{\partial v}{\partial t} = -\frac{\partial p}{\partial x}, \tag{4.9}$$

where p and v remain as the pressure and velocity variables respectively. The Hermite-Leapfrog schemes discretize the first order system by staggering pressure and velocity approximations in both space and time. An N^{th} order Hermite-Leapfrog method initiates the solution of the pressure variable at the initial time, t_0, on a primary grid using the function value and first N derivatives. Here the primary grid is defined in the usual manner

$$\Omega = \{x_m : \quad x_m = x_{min} + mh_x, \quad m = 0, \ldots, K-1\}.$$

The solution of the velocity variable is initiated on a dual grid, $\tilde{\Omega}$,

$$\tilde{\Omega} = \left\{x_{m+1/2} = x_{min} + (m+1/2)\, h_x, \quad m = 0, \ldots, K-1\right\},$$

at time $t_0 + \Delta t/2$. Similarly the velocity variable is represented using the function value and first N derivatives on each node of the dual grid. The evolution of the degrees of freedom is carried out using the update formula

$$\frac{\partial^n p(x, t + \Delta t)}{\partial x^n} - \frac{\partial^n p(x, t - \Delta t)}{\partial x^n} = \tag{4.10}$$
$$2 \sum_{l=0} \frac{1}{2l!} \left(\frac{\Delta t}{2} \right)^{2l} \frac{\partial^{2l+n} p(x,t)}{\partial t^{2l} \partial x^n} = -2 \sum_{l=0} \frac{1}{2l!} \left(\frac{\Delta t}{2} \right)^{2l} \frac{\partial^{2l+n} v(x,t)}{\partial x^{2l+n}},$$

where n corresponds to the order of the derivative. The last term of Equation 4.10 is the result of applying the Cauchy-Kowelasky recurrence relation to Equation 4.7. Noticeably, this allows the pressure variable to be evolved using the spatial derivatives of the velocity term. The spatial derivatives are approximated by means of Hermite interpolation. The evolution is then carried out so that all the terms are included. Figure 4.1 provides an illustration of the stencil associated with updating the pressure variable. The complete update formula for each variable is given by

$$\frac{\partial^n p(x, t + \Delta t)}{\partial x^n} = \frac{\partial^n p(x, t - \Delta t)}{\partial x^n} - \sum_{j=1, odd} 2 \frac{(\Delta t/2)^j}{j!} \frac{\partial^{j+n} v(x, t + \Delta t/2)}{\partial x^{j+n}}, \tag{4.11}$$

and

$$\frac{\partial^n v(x, t + 3\Delta t/2)}{\partial x^n} = \frac{\partial^n v(x, t + \Delta t/2)}{\partial x^n} - \sum_{j=1, odd} 2 \frac{(\Delta t/2)^j}{j!} \frac{\partial^{j+n} p(x, t + \Delta t)}{\partial x^{j+n}}. \tag{4.12}$$

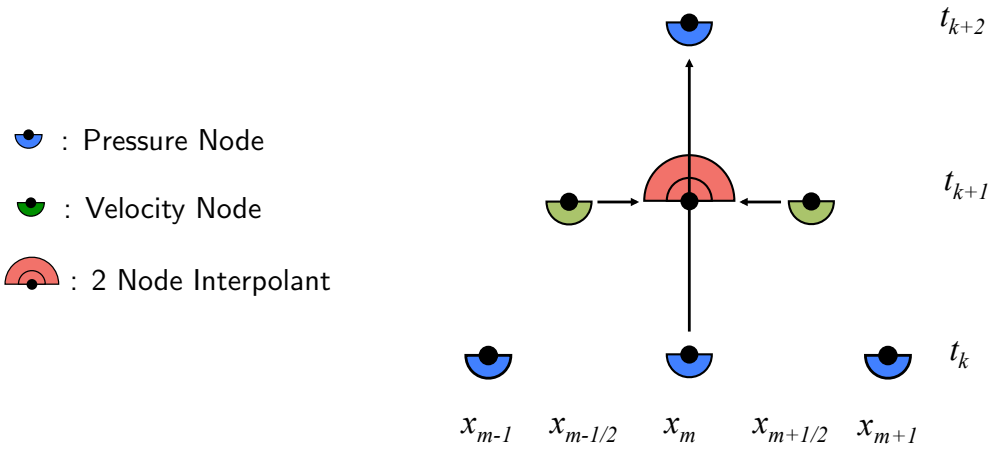


Figure 4.1 : Hermite-Leapfrog stencil for the acoustic wave equations as a pressure-velocity system.

4.3.2 Second Order Wave Equation

In one dimension the second order acoustic wave equation is expressed as

$$\frac{\partial^2 p}{\partial t^2} = \frac{\partial^2 p}{\partial x^2}. \tag{4.13}$$

Here the Hermite-Leapfrog scheme discretizes the PDE by representing the pressure solution on a primary grid and dual grid separated by a time step $\Delta t/2$. The degrees of freedom are evolved by applying the following update formula:

$$\frac{\partial^n p(x, t + \Delta t/2)}{\partial x^n} + \frac{\partial^n p(x, t - \Delta t/2)}{\partial x^n} = \tag{4.14}$$
$$2 \sum_{l=0} \frac{1}{2l!} \left(\frac{\Delta t}{2} \right)^{2l} \frac{\partial^{2l+n} p(x, t_n)}{\partial t^{2l} \partial x^n} = 2 \sum_{l=0} \frac{1}{2l!} \left(\frac{\Delta t}{2} \right)^{2l} \frac{\partial^{2l+n} p(x, t_k)}{\partial x^{2l+n}}.$$

This formula is derived from Equation 4.8. Here n corresponds to the order of the derivative. The last term in Equation 4.14 is the result of exchanging time derivatives for space derivatives by means of the Cauchy-Kowelasky recurrence relation. As

before the spatial derivatives on the right hand side are approximated by the Hermite interpolant. Figure 4.2 illustrates the one-dimensional stencil.

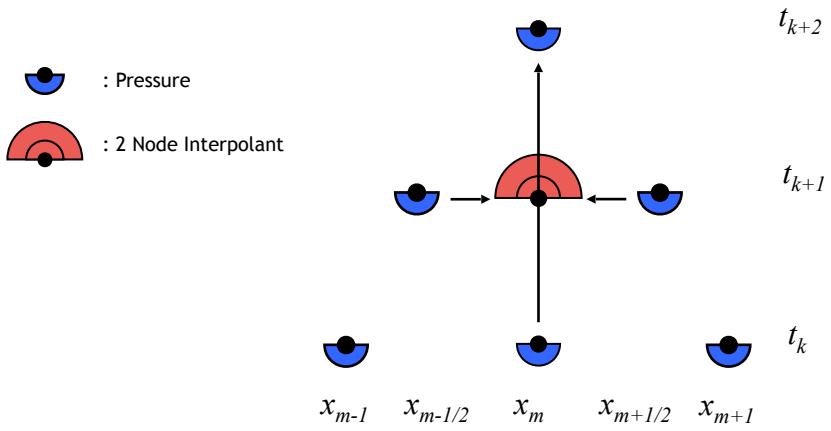


Figure 4.2 : Hermite-Leapfrog stencil for the second order acoustic wave equation.

4.3.3 The Method in Three-dimensions

Each Hermite-Leapfrog scheme may be extended to higher dimensions by means of a tensor product construction. For each node on the primary grid, $(x_m, y_m, z_m) \in \Omega$, a temporal Taylor series is constructed centered at $t = t_k + \frac{\Delta t}{2}$ and $t = t_k - \frac{\Delta t}{2}$

$$\frac{\partial^{n+l+s} p(x,y,z,t+\Delta t/2)}{\partial x^n \partial y^l \partial z^s} = \sum_{r=0} \frac{1}{r!} (\Delta t/2)^r \frac{\partial^{r+n+l+s} p(x,y,z,t_k)}{\partial t^r \partial x^n \partial y^l \partial z^s} \tag{4.15}$$

$$\frac{\partial^{n+l+s} p(x,y,z,t-\Delta t/2)}{\partial x^n \partial y^l \partial z^s} = \sum_{r=0} \frac{1}{r!} (\Delta t/2)^r \frac{\partial^{r+n+l+s} p(x,y,z,t_k)}{\partial t^r \partial x^n \partial y^l \partial z^s}. \tag{4.16}$$

Here the indeces n, l, s correspond to the order of spatial derivatives in a given dimension. Analogous to the one-dimensional version, an update scheme for the pressure term in the pressure-velocity system is obtained by subtracting Equations 4.15 and

4.16:

$$\frac{\partial^{n+l+s} p(x,y,z,t+\Delta t/2)}{\partial x^n \partial y^l \partial z^s} - \frac{\partial^{n+l+s} p(x,y,z,t-\Delta t/2)}{\partial x^n \partial y^l \partial z^s} = \tag{4.17}$$
$$- \sum_{r=1,odd} \frac{1}{r!} (\Delta t/2)^r \frac{\partial^{r+n+l+s} v_x(x,y,z,t)}{\partial x^{n+r} \partial y^{l+r} \partial z^{s+r}}$$
$$- \sum_{r=1,odd} \frac{1}{r!} (\Delta t/2)^r \frac{\partial^{r+n+l+s} v_y(x,y,z,t)}{\partial x^{n+r} \partial y^{l+r} \partial z^{s+r}}$$
$$- \sum_{r=1,odd} \frac{1}{r!} (\Delta t/2)^r \frac{\partial^{r+n+l+s} v_z(x,y,z,t)}{\partial x^{n+r} \partial y^{l+r} \partial z^{s+r}},$$

where the right-hand side is the result of exchanging the time derivatives for spatial derivatives via the Cauchy-Kowalsky recurrence relation. Constructing the update for the three-dimensional second order wave equation is carried out by adding Equations 4.15 and 4.16

$$\frac{\partial^{n+l+s} p(x,y,z,t+\Delta t/2)}{\partial x^n \partial y^l \partial z^s} + \frac{\partial^{n+l+s} p(x,y,t-\Delta t/2)}{\partial x^n \partial y^l \partial y^s} = \tag{4.18}$$
$$2 \sum_{r=0} \frac{1}{2r!} \left(\frac{\Delta t}{2} \right)^{2r} \frac{\partial^{n+l+s}}{\partial x^n \partial y^l \partial z^s} \Delta^r u(x,y,z,t_k).$$

Spatial derivatives of the right-hand side are again approximated by means of Hermite interpolation.

4.4 Recovering the Yee Scheme

The Yee scheme as introduced by Kane S. Yee, is a space-time staggered finite difference method for Maxwell's equations [25]. The scheme is easy to implement and fairly efficient for structured grids. The drawback of the scheme is that it is at best second order accurate. In this section I illustrate that the Yee scheme is a special case of the Hermite-Leapfrog scheme.

The Hermite-Leapfrog scheme of order $N = 0$ for the pressure-velocity system approximates derivatives by difference approximations of the grid functions. The temporal discretization for the one-dimensional wave equation is given by:

$$p_m^{k+1} = p_m^k - \frac{\Delta t}{2} \frac{\partial v_m^{k+1/2}}{\partial x} \tag{4.19}$$

and

$$v_{m+1/2}^{k+3/2} = v_{m+1/2}^{k+1/2} - \frac{\Delta t}{2} \frac{\partial p_{m+1/2}^{n+1}}{\partial x}. \tag{4.20}$$

Here p_m^{k+1} corresponds to a grid function approximation at time $t_k = t_0 + \Delta t k$ at the m^{th} grid point. The Yee scheme approximates the first derivative of the right-hand side of each equation by the difference formula

$$\frac{\partial v}{\partial x} = \frac{0.5v(x + \Delta x/2, t) - 0.5v(x - \Delta x/2, t)}{\Delta x}. \tag{4.21}$$

The coefficients are identical to the coefficients used by the Hermite interpolant to approximate the first derivative.

4.5 Conservation of Energy

The "Yee scheme" falls under a class of integrators referred to as sympletic. Sympletic integrators have two main appealing qualities. The first is that they are time reversible. Here time-reversibility means that if the solution has been propagated forward k steps it may be propagated backwards k steps to arrive at the same starting position. Second they are known for their ability to conserve energy. Sympletic integrators are known for their conservation of energy and are typically used in the

context of Hamiltonian dynamical systems [79].

By construction the Hermite-Leapfrog schemes maintain time reversibility, furthermore this scheme conserves energy. This analysis is carried out in Fourier space. For compactness the solution at a given time-step is expressed as $p^k = p(x, t + \Delta tk)$ and $v^{k+1/2} = v(x, t + \Delta tk/2)$. The Fourier transform and inversion formulas are given by

$$\hat{u}(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \exp(-ikx) u(x) dx, \tag{4.22}$$

and

$$u(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \exp(ikx) \hat{u}(k) dk. \tag{4.23}$$

Applying the transform formula to the pressure and velocity updates results in

$$\hat{p}^{n+1} = \hat{p}^n + \sum_{j=1,odd} 2 \frac{(\Delta tki/2)^j}{j!} \hat{v}^{n+1/2} \tag{4.24}$$

$$\hat{v}^{n+3/2} = \hat{v}^{n+1/2} + \sum_{j=1,odd} 2 \frac{(\Delta tki/2)^j}{j!} \hat{p}^{n+1}. \tag{4.25}$$

The series in Equations 4.24 and 4.25 may be simpled to $2i \sin\theta$, where $\theta = \frac{k\Delta t}{2}$, thereby simplifying the evolution procedure as the application of the following linear operator

$$\mathbf{S} = \begin{bmatrix} 1 & 2i \sin\theta \\ 2i \sin\theta & 1 - 4\sin^2\theta \end{bmatrix}.$$

As a result the evolution procedure is compactly denoted as

$$\begin{bmatrix} \hat{p}^{n+1} \\ \hat{v}^{n+3/2} \end{bmatrix} = \begin{bmatrix} 1 & 2i \sin\theta \\ 2i \sin\theta & 1 - 4\sin^2\theta \end{bmatrix} \begin{bmatrix} \hat{p}^n \\ \hat{v}^{n+1/2} \end{bmatrix}. \tag{4.26}$$

To gain insight into the behavior of the update matrix, $\mathbf{S}$, the eigenpairs of the matrix are computed. The eigenvalues are determined by computing the roots of the characteristic polynomial

$$\lambda^2 - \lambda(2 - 4\sin^2\theta) + 1,$$

which are simplified to be

$$\begin{aligned} \lambda_{1,2} &= 1 - 2\sin^2\theta \pm 2i\cos\theta\sin\theta \\ &= \cos 2\theta \pm i\sin 2\theta \\ &= \exp\left(\pm 2i\theta\right). \end{aligned} \tag{4.27}$$

The accompanying eigenvectors are determined to be:

$$\mathbf{v}_1 = det(\mathbf{V}) \begin{bmatrix} \exp(-i\theta) \\ 1 \end{bmatrix}, \quad \mathbf{v}_2 = det(\mathbf{V}) \begin{bmatrix} -\exp(i\theta) \\ 1 \end{bmatrix}. \tag{4.28}$$

I introduce $\mathbf{V}$ to be $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2]$, which leads to the diagonalization of

$$\mathbf{SV} = \mathbf{V} \begin{bmatrix} \exp(2i\theta) & 0 \\ 0 & \exp(-2i\theta) \end{bmatrix}. \tag{4.29}$$

Notably the operator $\mathbf{S}$ has unit eigenvalues. By defining

$$\mathbf{q}^n = \mathbf{V}^{-1} \begin{bmatrix} \hat{u}^n \\ \hat{v}^{n+1/2} \end{bmatrix}$$

allows the evolution to be expressed with the computed diagonalization:

$$\mathbf{q}^{n+1} = \mathbf{\Lambda}\mathbf{q}^{n}. \tag{4.30}$$

Since Λ is unitary the energy is conserved in the L^2 norm

$$\|\mathbf{q}^{n+1}\| = \|\mathbf{q}^{n}\|, \tag{4.31}$$

thus by Parseval's identity, the energy of the original system is conserved.

4.6 Numerical Experiments in One-Dimension

To assess the accuracy and behavior of the Hermite-Leapfrog schemes a series of numerical experiments are presented using the acoustic wave equations as a pressure-velocity system and as a second order equation. As in the previous chapter a CFL constant is introduced in order to specify the time step $\Delta t = 2C_{CFL}\frac{h}{c_{max}}$. Here h denotes the length of the interpolation interval and c_{max} denotes the maximum speed of the propagating wave. Numerical experiments were carried out in one dimension with constant and variable coefficients. Experiments with constant coefficients are then extended to three-dimensions. Numerical experiments suggest the Hermite-Leapfrog schemes can maintain a time-step independent of order.

4.6.1 Experiments with the Pressure-Velocity System

In order to introduce a spatially varying wave speed the function $c(x)$ is introduced to the one-dimensional wave equation

$$\begin{aligned} \frac{\partial p}{\partial t} = -c^2 \frac{\partial v}{\partial x} + h(x,t), \quad \frac{\partial v}{\partial t} = -\frac{\partial p}{\partial x} \\ p(x,t_0) = f(x), \quad v(x,t_0 + \Delta t/2) = g(x). \end{aligned} \tag{4.32}$$

The function $h(x,t)$ is then introduced in order to enforce a desired solution.

Standing Wave Solution

I begin by considering the propagation of a standing wave with unit wave speed. The analytic solution is chosen to be

$$p(x,t) = \cos(2\pi t)\sin(2\pi x).$$

For these experiments the computational domain is defined to be the bi-unit domain and the solution is propagated to a final time of $T = 4.13$. Figure 4.3 reports the accuracy in the L^2 norm while Table 4.1 reports the observed rates of convergence. Numerically the scheme appears to be stable. The experiments suggest that if the method is of even order one may expect rates of $O(h^{2N+2})$ while some variation is observed if the method is odd. Furthermore it can be observed that the Hermite-Leapfrog scheme can provide a better approximation than the classic Dual Hermite method when using choosing the order N to be even.

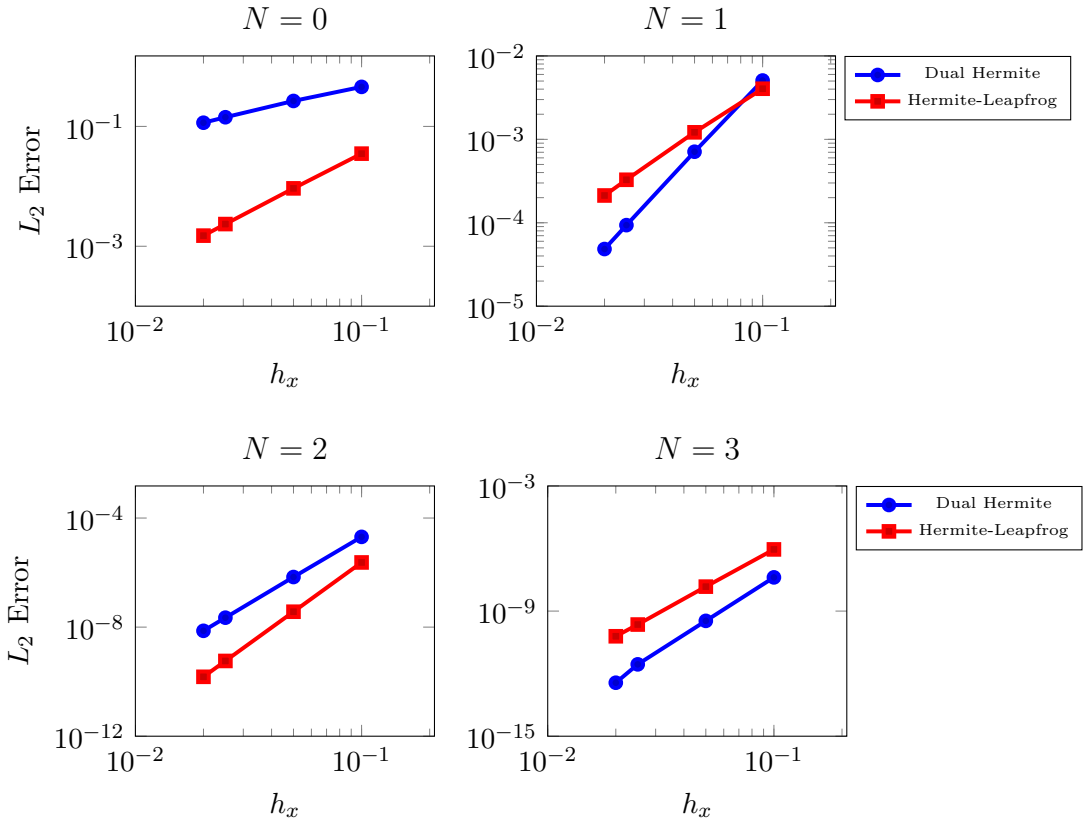


Figure 4.3 : L^2 errors of various N^{th} order Hermite-Leapfrog and Dual Hermite methods when applied to the one-dimensional pressure-velocity system. The Hermite-Leapfrog scheme provides a better approximation to the Dual-Hermite method when N is chosen to be even. For these experiments, the CFL constant is set to $CFL = 0.9$.

.

	$C_{CFL} = 0.1$				$C_{CFL} = 0.5$				$C_{CFL} = 0.9$			
Order - N	0	1	2	3	0	1	2	3	0	1	2	3
Hermite-Leapfrog	1.90	1.96	5.97	5.65	1.94	1.91	5.98	7.09	2.01	2.02	6.00	5.87
Dual Hermite	-	2.58	4.90	6.84	0.10	2.78	4.83	6.95	0.46	2.78	4.83	6.88

Table 4.1 : Observed L^2 rates of convergence for N^{th} order Hermite-Leapfrog and Dual Hermite methods with varying CFL constants.

Smoothly Varying Coefficients

Incorporating spatially varying coefficients to the Hermite-Leapfrog schemes is carried out in the same manner as discussed in Section 3.5.2. In the following set of numerical

experiments local Taylor series expansions of the coefficients,

$$c^2(x) = 1 + \sin(x)/2,$$

are computed at each grid point to order $2N + 1$. For these experiments the domain is chosen to be $[0, 2\pi]$ and the solution is propagated to a final time of $T = 3.2$. The analytic solution chosen to be

$$p(x, t) = \sin(x - t). \tag{4.33}$$

Figure 4.4 reports the observed accuracy in the L^2 norm while Table 4.2 reports the observed rates of convergence. These numerical experiments suggest rates of $O(h^{2N+2})$ for even N and $O(h^{2N})$ for odd N.

	$C_{CFL} = 0.1$				$C_{CFL} = 0.5$				$C_{CFL} = 0.9$			
N	0	1	2	3	0	1	2	3	0	1	2	3
Hermite-Leapfrog	1.98	1.97	5.98	5.94	1.98	1.97	5.99	5.88	1.98	1.98	6.02	5.81
Dual-Hermite	0.18	2.97	4.99	6.84	0.70	2.99	5.01	7.01	0.88	2.99	5.00	6.99

Table 4.2 : Observed L^2 rates of convergence for various N^{th} order Hermite-Leapfrog and Dual Hermite methods when applied to the pressure-velocity system with smoothly varying coefficients. The numerical experiments are carried out for various CFL constants.

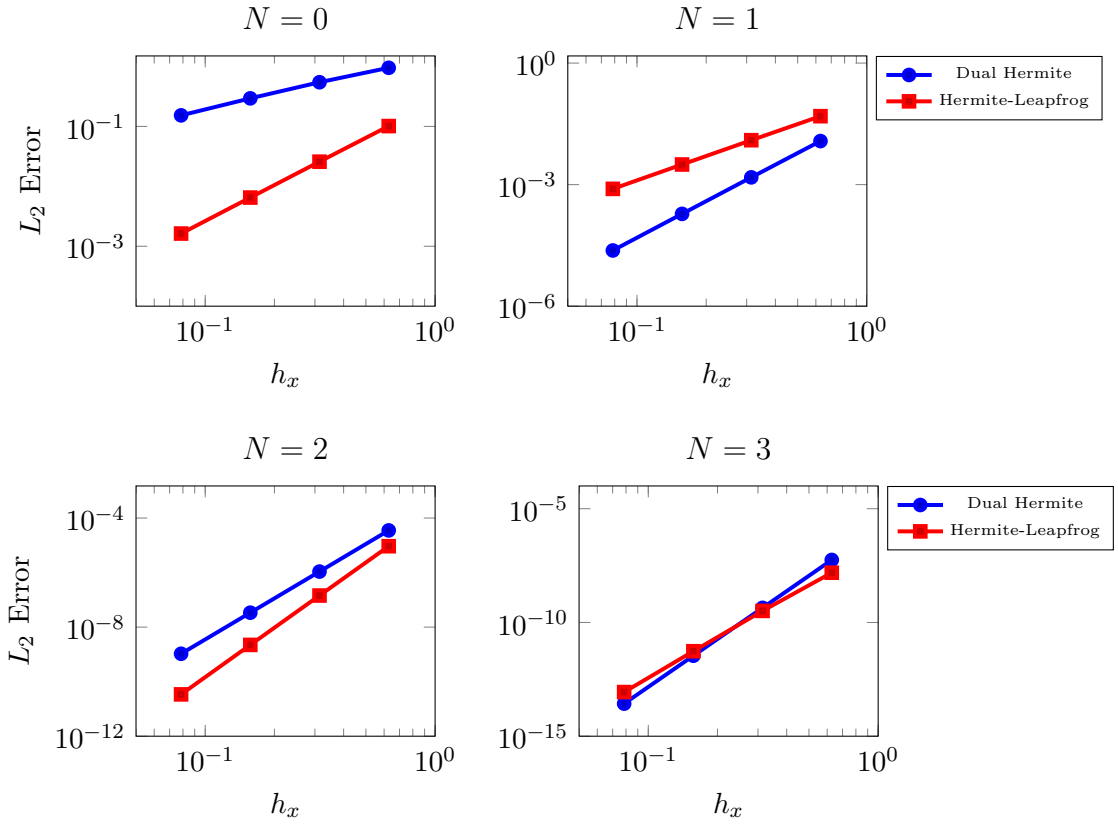


Figure 4.4 : The L^2 errors various N^{th} order Hermite-Leapfrog and Dual Hermite methods when applied to the one-dimensional pressure-velocity system with a spatially varying wave speed. The Hermite-Leapfrog scheme provides a better approximation than the Dual-Hermite method when N is chosen to be even. The CFL constant is set to $C_{CFL} = 0.9$.

.

4.6.2 Experiments with the Second Order Wave Equation

As a second set of numerical experiments I apply the Hermite-Leapfrog schemes to the second order acoustic wave equation

$$\frac{\partial^2 p}{\partial t^2} = c^2(x)\frac{\partial^2 p}{\partial x^2} + f(x,t) \tag{4.34}$$
$$p(x,0) = g_1(x), \quad \frac{\partial p(x,0)}{\partial t} = g_1(x).$$

Here the wave speed is governed by the function $c(x)$ and the $f(x,t)$ denotes a forcing term.

Standing Wave Solution

The numerical experiments are repeated with the standing wave solution for the second order acoustic wave equation with unit wave speed. The solution is propagated to a final time of $T = 4.13$. Figure 4.5 reports the observed errors while Table 4.3 reports observed rates of convergence. The numerical experiments suggest a convergence rate of $O(h^{2N})$. Furthermore taking too small of a time step results in under resolving the wave in the case of an order $N = 1$ scheme.

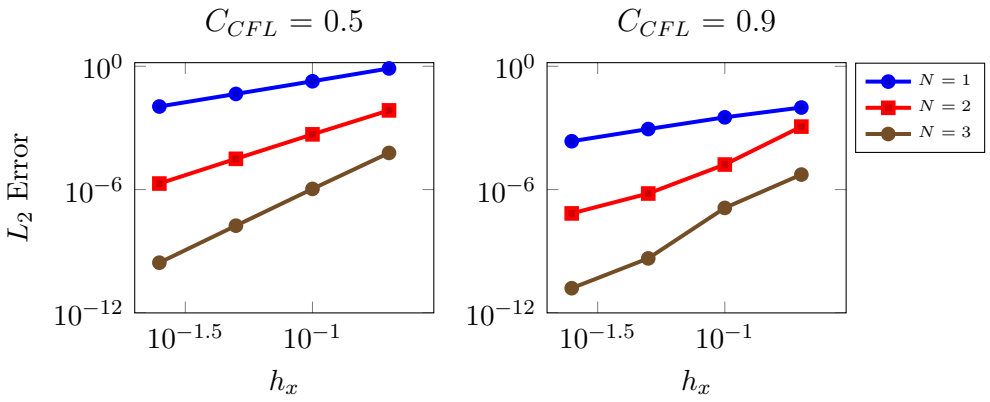


Figure 4.5 : The L^2 errors of various N^{th} order Hermite-Leapfrog schemes when applied to the second-order wave equation with CFL constants set to $C_{CFL} = 0.5$ and $C_{CFL} = 0.9$. The equation is set to have unit wave speed.

.

	$C_{CFL} = .1$			$C_{CFL} = .5$			$C_{CFL} = .9$		
Order - N	1	2	3	1	2	3	1	2	3
Hermite-Leapfrog	-	4.02	5.99	2.05	3.94	5.91	1.82	4.67	6.32

Table 4.3 : The L^2 rates of convergence for various N^{th} order Hermite-Leapfrog methods applied to the wave equation in the second order form with unit wave speed and various CFL cosntants.

Smoothly Varying Coefficients

The numerical experiments for variable coefficients are repeated using the same spatially varying wave speed

$$c^2(x) = 1 + \sin(x)/2.$$

The domain is chosen to be $[0, 2\pi]$ and the solution is propagated to a final time of $T = 4.5$. For these numerical experiments the analytic solution is chosen to be

$$p(x,t) = \cos((x-t)). \tag{4.35}$$

Figure 4.6 reports the accuracy of the method while Table 4.4 reports the observed rates of convergence for various CFL constants.

	$C_{CLF} = .1$			$C_{CLF} = .5$			$C_{CLF} = .9$		
Order - N	1	2	3	1	2	3	1	2	3
Hermite-Leapfrog	1.59	3.99	6.00	1.99	4.00	6.00	1.97	3.96	5.98

Table 4.4 : The L^2 rates of convergence for an N^{th} order Hermite-Leapfrog method when applied to the second order wave equation with smoothly varying coefficients. Different step sizes chosen by varying the CFL constant.

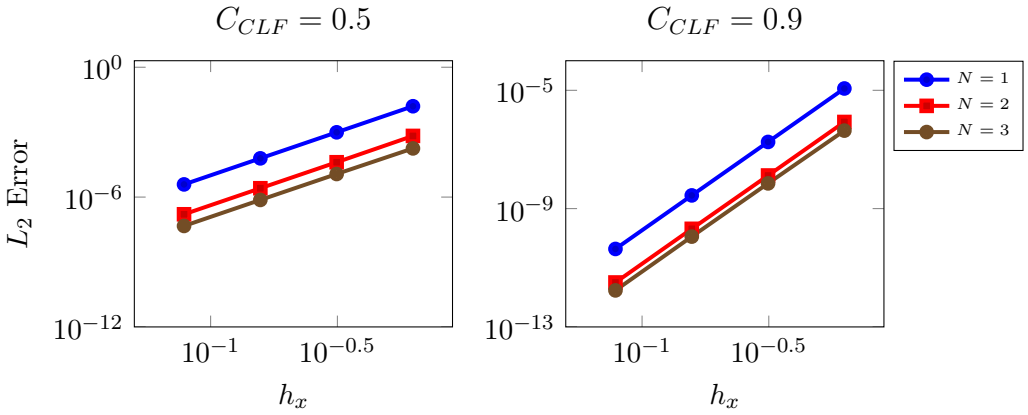
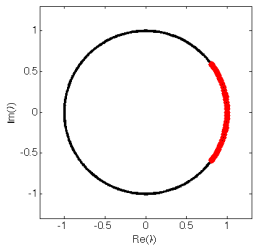


Figure 4.6 : The L^2 errors of various N^{th} order Hermite-Leapfrog schemes when applied to the second-order wave equation. Here the wave speed is assumed to be spatially varying. Furthermore different qualitative behavior may be observed by varying the CFL constant.

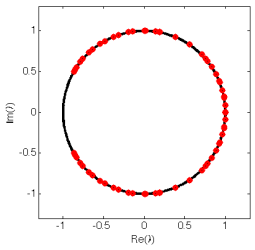
.

4.6.3 Spectra

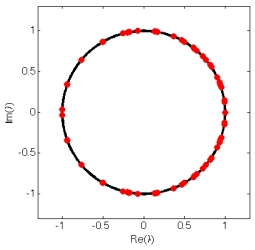
Hermite-Leapfrog methods also differ from the previously discussed schemes as evolution of the solution is maintained on the grid in which the solution is discretized on. Following the approach carried out in Section 3.5.3, Figure 4.7 illustrates the spectrum of an $N = 3$ method. Unlike Hermite-Taylor methods, the eigenvalues of the companion matrix are all found on the unit circle. Having the eigenvalues on the unit circle illustrates the schemes are non-dissipative. Furthermore, the eigenvalues clustered around the point (1,0) corresponds to the non-dissipative propagation of modes with small frequency. Similarly, the eigenvalues clustered around (-1,0) illustrate that high-frequency modes may be propagated without dissipation as well.



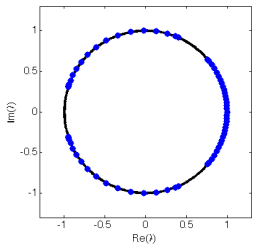
(a) Pressure-Velocity System, $C_{CFL} = .1$



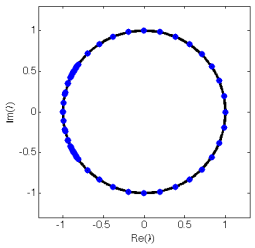
(b) Pressure-Velocity System, $C_{CFL} = .5$



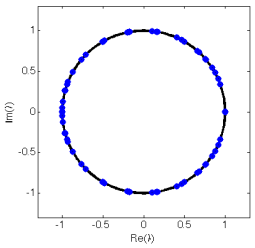
(c) Pressure-Velocity System, $C_{CFL} = .9$



(d) Second order Wave Equation, $C_{CFL} = .1$



(e) Second order Wave Equation, $C_{CFL} = .5$



(f) Second order Wave Equation, $C_{CFL} = .9$

Figure 4.7 : Spectra of the update matrix for the Hermite-Leapfrog scheme at various CFL constants. The order of approximation and number of cells are fixed to be $N = 3$ and $K = 16$, respectively. Noticeably the eigenvalues are found on the unit circle revealing the non-dissipative nature of the scheme.

4.7 Numerical Experiments in Three-Dimensions

For completeness, numerical experiments in three dimensions are carried out with both the first and second-order formulations of the wave equation. For each experiment the standing wave solution assumed

$$p(x, y, z, t) = \cos(2\pi t)\sin(2\pi x)\sin(2\pi y)\sin(2\pi z).$$

The domain is chosen to be $[-1, 1]^3$ and the solution is propagated to a final time of $T = 3.4$. The CFL constant is fixed at $C_{CFL} = 0.8$. Accuracy and observed rates of convergence are reported for the second order wave equation in Table 4.5. Al-

though higher than expected convergence rates are reported, numerical experiments demonstrated that by taking a smaller time step the scheme may achieve more consistent $O(h^{2N})$ convergence rates. Table 4.6 reports the observed accuracy and rates of convergence for the three dimensional pressure-velocity system.

	$N=1$		$N=2$		$N=3$	
h	L^2 Error	rate	L^2 Error	rate	L^2 Error	rate
0.2	0.000185596		0.000150751	-	1.44484e-07	-
0.1	0.000578585	-	4.03161e-07	8.54	5.3031e-10	8.09
0.05	0.000201805	1.56	7.60275e-08	2.40	2.55736e-11	4.37
0.025	4.975e-05	2.02	5.0191e-09	3.92	4.05316e-13	5.97

Table 4.5 : Observed L^2 errors and rates of convergence for N^{th} order Hermite-Leapfrog schemes when applied to the three-dimensional second order acoustic wave equation.

	$N=0$		$N=1$		$N=2$		$N=3$	
h	L^2 Error	rate	L^2 Error	rate	L^2 Error	rate	L^2 Error	rate
0.2	0.328171	-	0.00461116	-	5.23937e-05	-	1.0184e-07	-
0.1	0.534397	-	0.00128227	1.86	5.63095e-07	6.53	1.04121e-10	9.93
0.05	0.123504	2.11	0.000507941	1.33	9.83477e-09	5.83	7.41924e-12	4.37
0.025	0.0266204	2.21	0.000126843	2.00	1.53897e-10	5.99	-	

Table 4.6 : Observed L^2 errors and rates of convergence for the N^{th} order three-dimensional pressure-velocity system. Noticeably $O(h^{2N+2})$ convergence rates were observed for even N, this remains consistent with the one-dimensional experiments.

4.8 A Hybrid Hermite-Leapfrog Method

The peculiar convergence behavior of the Hermite-Leapfrog scheme, when applied to the pressure-velocity system, motivates the need for a local truncation analysis. As a starting point I discretize the advection equation

$$u_t = u_x, \tag{4.36}$$

using the Hermite-Leapfrog schemes and trace out local truncation errors. My goal here is to improve on accuracy and derive a scheme with more consistent rates of convergence. As a starting point, I am interested in comparing the truncation errors of the Hermite-Leapfrog schemes. For clarity, I carry out this analysis using an $N = 1$ scheme. An $N = 1$ scheme reconstructs the function value and first three derivatives at the midpoint of cells. By subtracting Equations 4.6a and 4.6b the resulting updates and their truncation errors are given by

$$u^{k+1} = u^k + \frac{\Delta t}{2}\frac{\partial u^{k+1/2}}{\partial x} + \frac{(\Delta t/2)^3}{3!}\frac{\partial^3 u^{k+1/2}}{\partial x^3} + O(\Delta t^5), \tag{4.37}$$

$$\frac{\partial u^{k+1}}{\partial x} = \frac{\partial u^k}{\partial x} + (\Delta t/2)\frac{\partial^2 u^{k+1/2}}{\partial x^2} + O(\Delta t^3). \tag{4.38}$$

Furthermore, by adding Equations 4.6a and 4.6b the resulting updates and their truncation errors are given by

$$u^{k+1} = -u^k + u^{n+1/2} + \frac{(\Delta t/2)^2}{2!}\frac{\partial^2 u^{n+1/2}}{\partial x^2} + O(\Delta t^4). \tag{4.39}$$

$$\frac{\partial u^{n+1}}{\partial x} = -\frac{\partial u^n}{\partial x} + \frac{\partial u^{n+1/2}}{\partial x} + \frac{(\Delta t/2)^2}{2!}\frac{\partial^3 u^{n+1/2}}{\partial x^3} + O(\Delta t^4). \tag{4.40}$$

From this simple truncation analysis it can be observed that the approximation for the function value produced by subtracting the Taylor series (Equation 4.37) is more accurate than the approximation produced by adding the Taylor series (Equation 4.39). Furthermore it is the derivative approximation produced by adding the Taylor series that yields a better approximate. Building on this observation I propose using Equation 4.37 to evolve the function value and Equation 4.40 to evolve the first derivative. This introduced the hybrid Hermite-Leapfrog scheme where the updates are

$$u^{n+1} = u^n + (\Delta t/2)\frac{\partial u^{n+1/2}}{\partial x} + \frac{(\Delta t/2)^3}{3!}\frac{\partial^3 u^{n+1/2}}{\partial x^3} \tag{4.41}$$

$$\frac{\partial u^{n+1}}{\partial x} = -\frac{\partial u^n}{\partial x} + (\Delta t/2)\frac{\partial u^{n+1/2}}{\partial x} + \frac{(\Delta t/2)^2}{2!}\frac{\partial^3 u^{n+1/2}}{\partial x^3}. \tag{4.42}$$

To better understand the amplification of error the next section studies the approximation error introduced by the interpolation from a point-wise perspective.

4.8.1 Interpolation Operator via Taylor Series

The construction of the Hermite interpolation operator as described by Goodrich in [15] uses Hermite-Lagrange basis function. The interpolation error of the polynomial in the L^2 sense is then derived through the use of the Peano kernel representation. It is pointed out in their analysis that a Hermite interpolant approximates a function at a rate of $O(h^{2N+2})$ over a local cell of width h. Furthermore, with each spatial differentiation, the polynomial looses an order of approximation. Thus the rate of convergence is $O(h^{2N+2-r})$ where r corresponds to the order of the derivative being

considered.

In this section I am interested in the point-wise accuracy of the interpolation operator. To estimate point-wise accuracy I present an alternative derivation of the interpolation operator based on local Taylor series expansions. At an arbitrary grid point, $x_{m+1/2}$, I consider a Taylor series expansion at neighboring grid points $x_{m\pm1}$, with $h/2$ grid spacing

$$u(x \pm h/2, t) = u \pm \frac{h}{2}u^{(1)} + \frac{(h/2)^2}{2!}u^{(2)} \pm \frac{(h/2)^3}{3!}u^{(3)} + O(h^4) \tag{4.43}$$

$$u^{(1)}(x \pm h/2, t) = u^{(1)} \pm \frac{h}{2}u^{(2)} + \frac{(h/2)^2}{2!}u^{(3)} \pm O(h^3). \tag{4.44}$$

Here the subscripts correspond to the order of the spatial derivative. This type of expansion introduces four linear equations with four unknowns, namely the function value and derivatives at the midpoint $x_{m+1/2}$. The function values and first $2N+1$ derivatives may then be computed by solving the following linear system:

$$\begin{pmatrix} 1 & -h/2 & \frac{(h/2)^2}{2!} & -\frac{(h/2)^3}{3!} \\ 0 & 1 & -h/2 & \frac{(h/2)^2}{2!} \\ 1 & h/2 & \frac{(h/2)^2}{2!} & \frac{(h/2)^3}{3!} \\ 0 & 1 & -h/2 & \frac{(h/2)^2}{2!} \end{pmatrix} \begin{pmatrix} u \\ u^{(1)} \\ u^{(2)} \\ u^{(3)} \end{pmatrix} = \begin{pmatrix} u(x-h/2) \\ u^{(1)}(x-h/2) \\ u(x+h/2) \\ u^{(1)}(x+h/2) \end{pmatrix}. \tag{4.45}$$

The inverse of the matrix provides the coefficients needed to approximate the function value and first $2N+1$ derivatives at the midpoint, $x_{m+1/2}$. A local truncation analysis demonstrates that the function value and first derivative are approximated at a rate of $O(h^4)$ while the subsequent derivatives are approximated at a rate of $O(h^2)$. Numerically this can be verified by considering the point-wise errors under

the L^2 norm

$$\|D^i u_a - D^i u_h\|_{L^2} = \sqrt{\sum_{j=0} \left(D^i u_a(x_j) - D^i u_h(x_j)\right)^2}.$$

Here the analytic solution is represented by u_a while the approximated solution is represented by u_h. The term D^i corresponds to the derivative operator computing the i^{th} spatial derivative. Numerical experiments further support this claim. Numerical experiments are carried out over the computational domain $[-2, 2]$. For these numerical experiments Equations 4.45 are used to approximate the function value and first $2N + 1$ derivatives of $\sin(2\pi x)$ at the midpoint of $\{15, 25, 35, \ldots, 75\}$ cells with equal width on the domain of $[-2, 2]$. Figure 4.8 reports the observed l_2 errors for the reconstructed function values and Table 4.7 reports the observed rates of convergence. Numerically it can be observed that the function value and first derivative are approximated at a rate of $O(h^4)$, while the second and third derivatives are approximated at a rate of $O(h^2)$. This behavior is expected as the coefficients used by the Hermite interpolation operator are central difference approximations which converge at even orders.

	$dx = 0$		$dx = 1$		$dx = 2$		$dx = 3$	
h	l_2 Error	rate	l_2 Error	rate	l_2 Error	rate	l_2 Error	rate
0.26	0.0195	-	0.0040	-	0.11293	-	0.06844	-
0.16	0.0026	3.94	5.20e-04	3.95	0.0416	1.95	0.0250	1.96
0.114	6.8e-04	3.97	1.37e-04	3.98	0.0213	1.98	0.0128	1.98
0.08	2.5e-04	3.98	5.04e-05	3.99	0.0129	1.98	0.0078	1.99
0.072	1.1e-04	3.99	2.26e-05	3.99	0.0087	1.99	0.0052	1.99
0.061	5.00e-05	3.99	1.16e-05	3.99	0.0062	1.99	0.0037	1.99

Table 4.7 : As a complement to Figure 4.8 this table reports the observed point-wise accuracy under the the l_2 norm and rates of convergence in approximating spatial derivatives, dx, using Hermite interpolation.

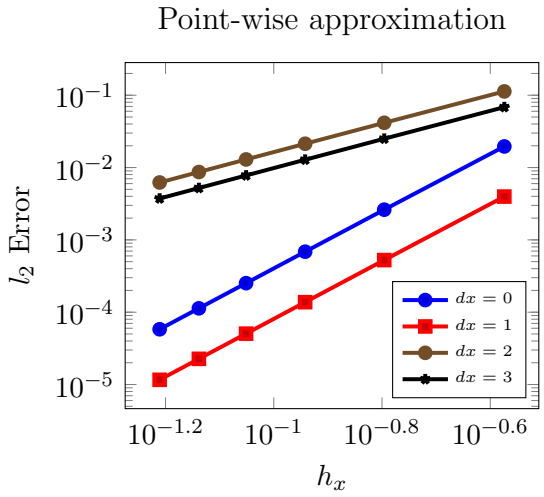


Figure 4.8 : Observed l_2 errors when approximating the function value and first three derivatives of a sine function on a collection of cells of width h_x for the domain $[-2, 2]$ via Hermite interpolation. Here the function value and first derivatives at vertices of cells are being used to approximate the data at the midpoint.

4.8.2 Amplification of Error

In addition to the truncation of the temporal series, the error is further amplified by the approximation of function and derivatives values. Building on the local truncation analysis and numerical experiments observed in Section 4.8.1, the errors in Equations 4.41 and 4.42 are furthered amplified in the following manner

$$u^{k+1} = u^{k-1} + \sum_{l=odd} c_l \Delta t^l \frac{\partial^l \tilde{u}^k}{\partial x^l} + O\left(\alpha_1 h^4 \Delta t + \alpha_2 h^2 \Delta t^3 + \alpha_3 \Delta t^5\right) \tag{4.46}$$

$$\frac{\partial u}{\partial x}^{n+1} = -\frac{\partial u}{\partial x}^{n} + \sum_{l=even} d_l \Delta t^l \frac{\partial^{l+1} \tilde{u}}{\partial x^{l+1}} + O\left(\alpha_1 h^4 + \alpha_2 \Delta t^2 h^2 + \Delta t^4\right). \tag{4.47}$$

Here the variables with a tilde correspond to the approximated value. Assuming the scheme is numerically stable, I expect the Hybrid Hermite-Leapfrog scheme to convergence at a rate of $O(h^4)$.

4.8.3 Numerical Experiments

To assess the accuracy and performance of the hybrid Hermite-Leapfrog scheme numerical experiments are carried out using the one-dimensional advection equation. For these experiments the domain is chosen to be the bi-unit domain and the analytic solution is chosen to be

$$u(x,t) = \sin(3\pi(x-t)).$$

Furthermore I extend the Hybrid Hermite-Leapfrog method to orders $N = 2, 3$ and using similar arguments as in the case of $N = 1$, a convergence rate of $O(h^{2N+2})$ is expected. Figure 4.9 reports the observed accuracy for the rates while Table 4.8 reports the observed rates of convergence. Noticeably this scheme achieves consistent $O(h^{2N+2})$ convergence rates.

	$C_{CFL} = .1$			$C_{CFL} = .5$			$C_{CFL} = .9$		
Order - N	1	2	3	1	2	3	1	2	3
Hybrid Hermite	4.15	5.92	8.06	4.08	5.96	8.00	3.94	5.98	7.99

Table 4.8 : Observed L^2 errors and rates of convergence when using the Hybrid Hermite-Leapfrog scheme to solve the advection equation with unit wave speed.

Pressure-Velocity System in One-Dimension

The Hybrid Hermite-Leapfrog scheme may also be applied to the pressure-velocity system (Equation 4.32) but has the burden of requiring additional copies of the solution. In particular, it becomes necessary to have the pressure and velocity in both the primary and dual grids. The two copies are necessary since applying the Cauchy-Kowelasky recurrence relation to the pressure variable exchanges odd time derivatives

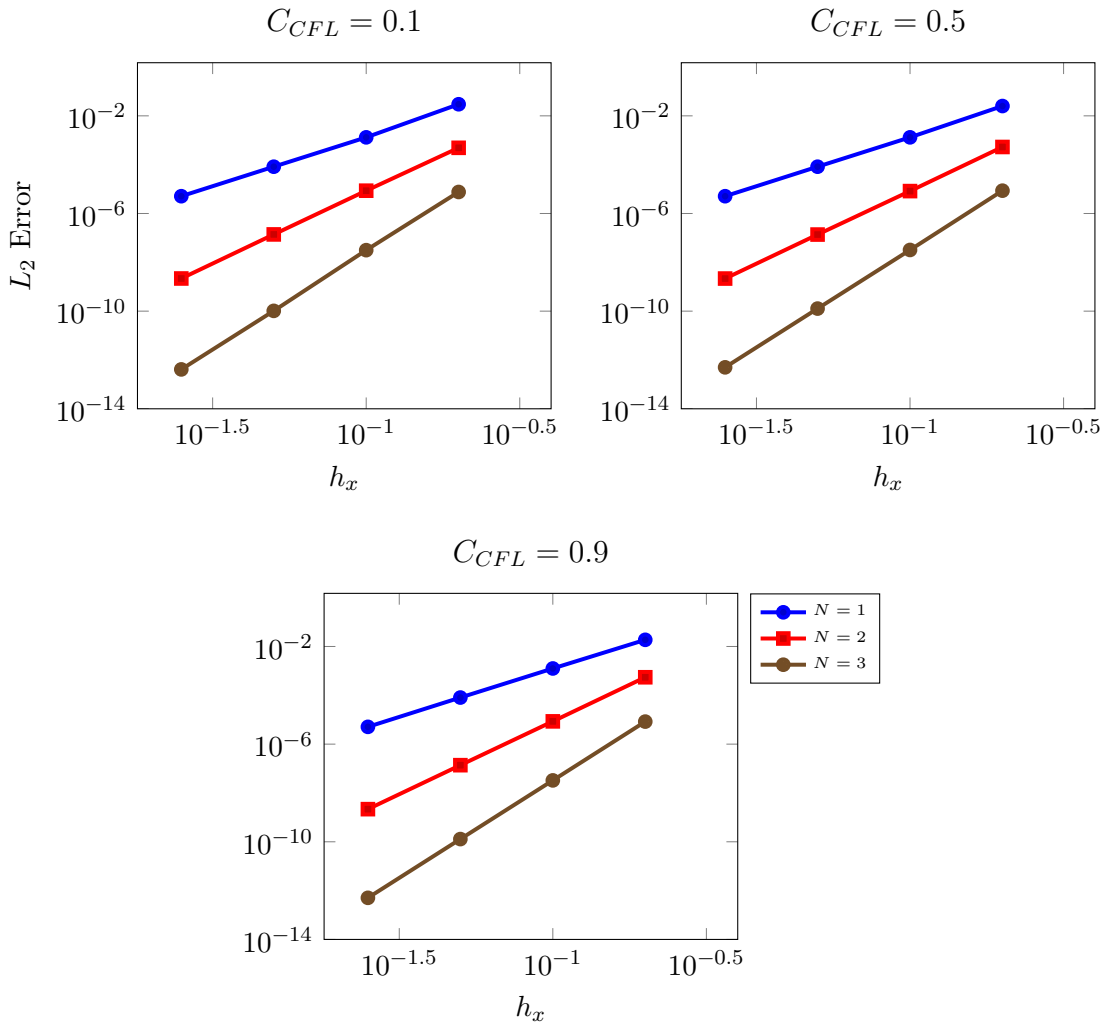


Figure 4.9 : Errors in the L^2 norm for an N^{th} order Hybrid Hermite-Leapfrog scheme assuming a CFL constant of $C_{CFL} = 0.9$. The advection equation is assumed to have unit wave speed.

.

of the pressure variable for odd spatial derivatives of the velocity variable, and even time derivatives of the pressure variable for even spatial derivatives of the pressure variable. For completeness, I compare all variations of Hermite-Leapfrog methods discussed in this chapter with the classic Dual Hermite Method introduced by Goodrich in [15].

For these numerical experiments the analytic solution for pressure term is chosen to be

$$p(x,t) = \cos(3\pi t)\sin(3\pi x),$$

and the solution is propagated to a final time of $T = 4.13$. Figure 4.10 reports the observed accuracy when the methods are applied to the one-dimensional acoustic wave equation. Table 4.9 reports the observed rates of convergence. The numerical experiments remain consistent with the $O(h^{2N+2})$ convergence rates observed with the advection equation. Furthermore, it is observed that the hybrid Hermite-Leapfrog scheme yields the best approximation in comparison to the standard Dual Hermite and Hermite-Leapfrog schemes.

	$C_{CFL} = .1$			$C_{CFL} = .5$			$C_{CFL} = .9$		
Order - N	1	2	3	1	2	3	1	2	3
Hybrid Hermite-Leapfrog	4.03	6.07	7.99	3.98	5.98	8.04	3.93	5.93	7.97
Hermite-Leapfrog 1^{st}	1.04	5.95	5.07	1.07	5.95	5.64	1.21	5.98	6.50
Hermite-Leapfrog 2^{nd}	0.73	3.97	5.98	1.72	3.89	5.85	1.81	3.86	6.09
Dual-Hermite	1.89	4.95	7.08	2.56	4.99	6.93	2.87	4.94	6.96

Table 4.9 : Comparison of L^2 rates of convergence for the N^{th} order Hybrid Hermite-Leapfrog, Hermite-Leapfrog variants, and the Dual Hermite methods when applied to the acoustic wave equations.

4.9 Summary

This chapter introduced Hermite methods which use leapfrog time-stepping. These Hermite-Leapfrog methods have the flexibility of being applied to either equations in first or second order form. Numerical experiments suggested that the Hermite-Leapfrog scheme, when applied to a second order equation, converges at a rate of $O(h^{2N})$ while some variation was found when applied to a first order equations. No-

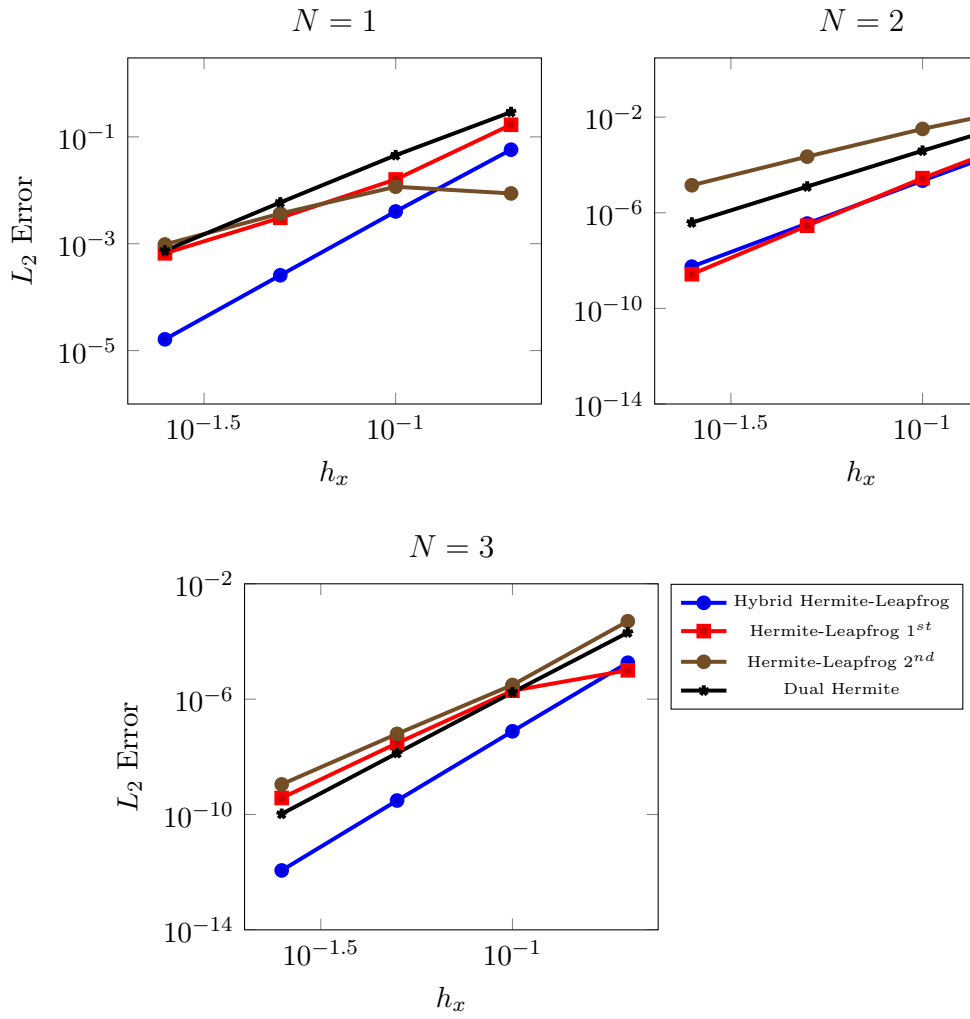


Figure 4.10 : L^2 errors for the Hybrid Hermite-Leapfrog scheme, Hermite-Leapfrog variants, and Dual Hermite scheme when applied to the acoustic wave equations.

.

tably when N was chosen to be even the method converged at consistent rates of $O(h^{2N+2})$. The variation occurred when N was chosen to be odd, the observed rate of convergence varied between $O(h^{2N})$ to $O(h^{2N+1})$. The observed variation of the Hermite-Leapfrog scheme when applied to the pressure velocity system motivated a local truncation analysis. By tracing out local truncation errors a hybrid Hermite-

Leapfrog scheme was derived which achieves consistent rates of $O(h^{2N+2})$. The work presented in this chapter was purely numerical and thus an immediate area of future research is investigating mathematical arguments for stability.

Chapter 5

GPU Accelerated Hermite Methods

In this chapter I expose parallelism in Hermite methods and explore the graphics processing unit in order to accelerate computations. Recent trends in processor design has resulted in multi-core processors with wide single instruction multiple data (SIMD) vector units. Each SIMD group has access to a relatively small shared memory cache and each SIMD lane has a small number of fast registers. Typical GPUs are further equipped with large bandwidth, high latency, global shared memory storage. To achieve high performance on GPUs fine-grained parallelism must be exposed with minimal communication between computing units. It is because the GPU exemplifies many/multi-core trend in computer architecture that I consider it as a platform for accelerating Hermite methods. For clarity and hardware portability numerical experiments are carried out through the use of the portable programming language OCCA [80].

5.1 The Graphics Processing Unit (GPU)

GPUs are computing devices originally designed to perform computations found in graphics rendering. Rather than relying on the general-purpose central processing unit (CPU), computations are offloaded onto graphics cards. Compared to the CPU, GPUs contain a much larger number of processor cores on a single chip and are optimized for single instruction multiple data parallelism. Each processor core on the

GPU is designed for simplified logic (limited cache, concurrent operations, no branch prediction) and in particular vectorized floating-point operations. GPUs achieve high performance by maintaining a high thread count to expose the fine-grained parallelism in the computation and the data movement.

5.1.1 GPU Application Program Interface

In 2007, with the release of NVIDIA's Compute Unified Device Architecture (CUDA), computational scientists were given access to simpler ways to program GPUs [81]. A drawback back of CUDA is that it is only designed to work with NVIDIA GPUs. As such, several industries came together to create a standardized API called the Open Compute Language (OpenCL) [81, 82]. The purpose of the language is to allow programmers to manage parallelism and data delivery in massive quantities through parallel processors. The OpenCL API goes beyond GPUs, allowing the programming model to be mapped to homogeneous or heterogeneous, single or multiple-device systems consisting of CPUs, GPUs, Field-Programmable Gate Arrays (FPGA), and potentially other future devices.

5.1.2 GPU Programming Model

OpenCL and CUDA are among the most popular APIs for developing GPU kernels. Both APIs employ the same memory hierarchy model. OpenCL however, allows the user to program a variety of devices but comes at the extra cost of having the user create a command queue, establish a relationship between the host (the hardware that will call the kernel), and the device (the hardware that will execute the kernel). A wrapper can easily facilitate writing multiple projects in OpenCL; further discussion on OpenCL can be found in [82]. NVIDIA's CUDA hides the host-device relationship

and provides a simpler environment in which to program GPUs at the cost of limiting programming to NVIDIA devices.

Computation on the GPU is performed on a predefined grid of compute units. Following NVIDIA's nomenclature each unit of the grid is referred to as a thread. Threads are grouped to form thread blocks. The hardware provides a similar hierarchy for memory. Threads are provided with a small amount of exclusive memory, threads in a thread block share block exclusive memory (shared memory), and lastly the entire compute grid shares global memory. Moving data between the CPU and GPU is accomplished through the use of global memory which acts as a general buffer. An in depth description regarding GPU computing may be found in [81].

The GPU experiments in this chapter are carried out using an NVIDIA GTX 980 GPU in single precision. The hardware has theoretical peak bandwidth of 224 GB/sec and a floating point performance of 4,612 GFLOP/sec. In practice however, these values tend to be difficult to achieve. A series of micro-benchmarks performed by Xinxin Mei and co-authors suggest that one can expect a streaming throughput of 156 GB/s [2] for memory bound kernels. Although a higher performance can be achieved, this thesis will treat these values to correspond as good performance numbers for kernels whose performance is limited by bandwidth.

5.2 Introduction to OCCA

In addition to GPUs, CPUs provide platforms for parallelism. CPU parallelism lends itself to various paradigms. One of the most popular models is based on message passing, where messages are passed through the CPU cores or a distributed network. The Message Passing Interface (MPI) is the standard API used in distributed computing. OpenMP offers a compiler directive approach for shared memory parallelism by em-

ploying a "fork-join model" in which a master thread forks into multiple threads to execute regions of code in parallel [83]. MPI may be paired with a threading language (OpenMP or a GPU API) to expose additional levels of parallelism [76]. The various threading APIs and the inability to predict lasting languages led to the development of OCCA [80]. OCCA brings flexibility to the programmer by providing a uniform programming interface for OpenMP, CUDA and OpenCL. At compile time OCCA translates key words in an OCCA kernel to language-specific words. OCCA has interfaces for various programming languages commonly used in scientific computing such as, C, C++, C#, Fortran, Matlab, Julia, and Python.

OCCA corresponds to a more general effort carried out by computational scientist to create a framework in which developers can establish hardware independence. Other examples of this type of efforts include RAJA as developed by scientist at Lawrence Livermore National Lab. RAJA serves as a C++ abstraction layer which allows developers to encapsulate platform specific concerns. The library focuses on kernel generation through functors relying on C++-11 and lambda functions [84]. Another example is KOKKOS developed at Sandia National laboratory. In contrast to OCCA and RAJA, KOKKOS uses multi-dimensional arrays and is at core a layered collection of C++ libraries. The library is designed to cover a variety of linear algebra routines for a variety of multi-threading APIs [85]. Although the inner workings of each API are different the development corresponds to a larger effort to maintain code portability.

OCCA maintains hardware portability by abstracting a GPU kernel as a series of nested for loops. An OCCA kernel is composed of "outer" and "inner" loops that are mapped to thread-blocks and threads in a thread block in the CUDA nomenclature. Inner loops may also be mapped to the OpenMP fork model in which inner loops

correspond to parallel regions. The strength of OCCA is its ability to provide a uniform framework in which the user may execute kernel codes in various languages. An effective OCCA implementation requires an underlying knowledge of the hardware, exposing parallelism, and the ability to tune kernels to each device. The listing bellow presents an example of an OCCA kernel.

```
kernel void myKernel(kFloat *U, kFloat *Uh, const kFloat *Hmatx){
  //Loop over blocks
  for(int outerId1 = 0; outerId1 < ny; outerId1++; outer1){
    for(int outerId0 = 0; outerId0 < nx; outerId0++; outer0){

      //Local threads
      for(int innerId1 = 0; innerId1 < innerMax1; innerId1++; inner1){
        for(int innerId0 = 0; innerId0 < innerMax0; innerId0++; inner0){

        }
      }

    }
  }

}
```

listings/myKernel.okl

5.3 Implementing Hermite Methods on the GPU

Exploiting the GPU architecture begins with a fundamental data structure. In this work multi-dimensional arrays (tensors) are used to hold the degrees of freedom of a Hermite discretization. For example in $\mathbb{R}^3$, the degrees of freedom are stored in as a rank 6 tensor

$$\mathbf{p}[m_3][m_2][m_1][n_3][n_2][n_1],$$

with n_1 being the fastest running index in memory. The data structure is used to store the function value and derivatives at each node of a three-dimensional grid.

The three innermost indices correspond to a node on the grid and the outermost indices catalog the corresponding tensor product of function value and derivatives. Polynomial reconstruction at a node on the dual grid, is accomplished by interpolating the function value and derivatives from vertices of the encapsulating cell. This requires reading $(N+1)^3$ degrees of freedom per vertex for a total of eight vertices in three dimensions (27 for the Virtual Hermite scheme). Regardless of the PDE being solved, the Hermite interpolation step appears in every solver and will serve as the starting point for tailoring Hermite methods onto the GPU.

As model equations I consider the acoustic wave equation as both the pressure-velocity system and as a single second order equation in three-dimensions. Performance studies are carried out using the Dual Hermite and Hermite-Leapfrog schemes. Kernel performance is evaluated by measuring effective arithmetic throughput and effective memory bandwidth through the NVIDIA profiler. The effective arithmetic throughput is computed using the number of floating-point operations given by flop_count_sp, and the effective memory bandwidth is obtained by summing *dram_read_throughput* and *dram_write_throughput*. Here bandwidth corresponds to the sum of bytes read and written to global memory by a GPU kernel.

5.3.1 Hermite Interpolation on the GPU

To facilitate the interpolation procedure a one-dimensional Hermite interpolation operator, $\mathbf{H}$ is pre-computed enabling dimension-by-dimension reconstruction of the polynomial. In this kind of reconstruction, the degrees of freedom of the encapsulating cell are stored in a local rank 3 tensor, $\mathbf{u_{loc}}$. The one-dimensional operator, $\mathbf{H}$, is then applied to the degrees of freedom of nodes parallel to the x_1 dimension as a series of matrix-matrix operations. Next, the operator is applied to the degrees of freedom

of nodes parallel to the x_2 dimension, and lastly to the degrees of freedom of nodes parallel to the x_3 dimension. For clarity we define $\mathbf{H}_{x_1}$, $\mathbf{H}_{x_2}$, and $\mathbf{H}_{x_3}$ as operators to be applied in the x_1, x_2, and x_3 dimensions respectively. Algorithm 2 presents the application of the interpolation operator to nodes parallel to the x_1 dimension using nested for loops. Applying the operator in the x_2, and x_3 dimensions is performed analogously. The complete reconstruction procedure for a single polynomial is listed as Algorithm 3.

Algorithm 2 Polynomial reconstruction in the x_1 dimension

```
1: procedure ReconstructionIn x1(Hx1, uloc, Ru)
2:     for tz=0:2N+1 do
3:         for ty=0:2N+1 do
4:             for tx=0:2N+1 do
5:                 c=0
6:                 for k=0:2N+1 do
7:                     c += Hx1[tx][k] uloc[tz][ty][k]
8:                 Ru[tz][ty][tx]=c;
```

Algorithm 3 Polynomial reconstruction

```
1: procedure PolynomialReconstruction(Hx1,Hx2,Hx3,uloc, Ru)
2:     Ru = Hx1 uloc
3:     uloc = Hx2 Ru
4:     Ru = Hx3 uloc
```

The GPU implementation exposes two levels of parallelism: coarse parallelism, in which threads in a block collectively reconstructs polynomials, and fine-grain parallelism in which threads carry out the local dot products found in matrix-matrix multiplications. The reconstruction is carried out locally by moving the necessary degrees of freedom to shared memory. By employing shared memory and a data structure which promotes coalesced reads/writes a kernel with basic optimizations

may be created. As optimizing a GPU kernel is an iterative process, five iterations of the kernel are described along with their performance gains. The first iteration of the kernel chooses thread block dimensions of $(N+1)^3$. Thus each thread in a given thread block is responsible for computing eight degrees of freedom of the interpolant. Initial performance profiling demonstrated that this approach left much room for improvement. A second iteration of the kernel exhibited a performance gain by increasing the number of threads per block to $(2N+2) \times (2N+2) \times (N+1)$ reducing the computational workload per thread.

To minimize and reuse global memory reads, the third iteration of the kernel employs a similar register rolling technique as used in Finite Difference Time Domain methods [1]. Hermite methods can mimic this technique by having a block of threads reuse a subset of shared memory. This is accomplished by setting up a two-dimensional grid of thread blocks. A single block of threads moves the bottom four vertices of a cell to shared memory. The block of threads then applies the interpolation operators $\mathbf{H}_{x_1}$ and $\mathbf{H}_{x_2}$. As it progresses along the x_3-dimension it stores the next four vertices of the cell in shared memory and applies $\mathbf{H}_{x_1}$ and $\mathbf{H}_{x_2}$ to the newly added degrees of freedom. As there are now degrees of freedom for eight vertices, the $\mathbf{H}_{x_3}$ operator is then applied to the degrees of freedom parallel to the x_3 dimension and the result is stored in a rank 6 tensor. The block of threads then shifts forward to the next set of four nodes and repeats the polynomial reconstruction. Figure 5.1 illustrate the shared memory rolling technique.

The fourth iteration of the kernel introduces a tunable parameter: the number of polynomials reconstructed along the x_1-dimension per block of threads. This further reduces the total amount of global memory reads as neighboring cells share nodes on the interface. The last optimization implemented is the introduction of the restrict

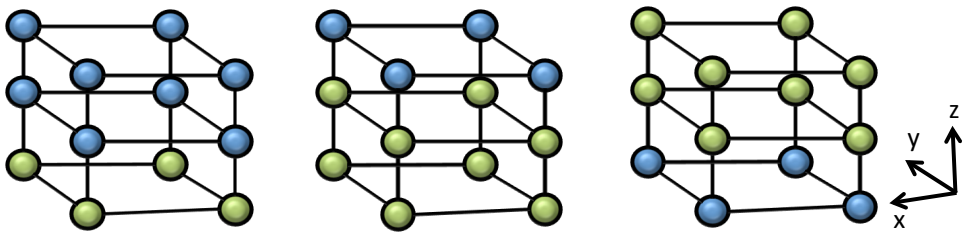


Figure 5.1 : Example of "Z-Looping" as introduced in [1], nodes in green correspond to data stored in a GPU's shared memory. As a block of threads marches through the Z-dimension data from global memory is moved to shared memory (blue to green).

qualifier and unrolling of for-loops. Comparing the effects of the various optimizations, Figure 5.2 presents the observed bandwidth and GFLOP as kernel optimizations are gradually accumulated. The following enumerated list corresponds to the gradual optimizations introduced to the interpolation kernel thereby complementing Figure 5.2. Finally as final metric, Table 5.1 presents the observed time to solution for each version of the Hermite Interpolation kernel.

1. Version 1: Threads per block are set to $(N+1)^3$ and shared memory is used perform the tensor operations.

2. Version 2: Threads per block are adjusted
   from $(N+1)^3$ to $(2N+1) \times (2N+1) \times (N+1)$.

3. Version 3: Thread blocks march through the slowest dimension of the domain and previously computed data is reused.

4. Version 4: A tuning parameter is introduced in which a block of threads constructs more than one interpolant.

5. Version 5: Restrict qualifiers are introduced and for-loops are unrolled.

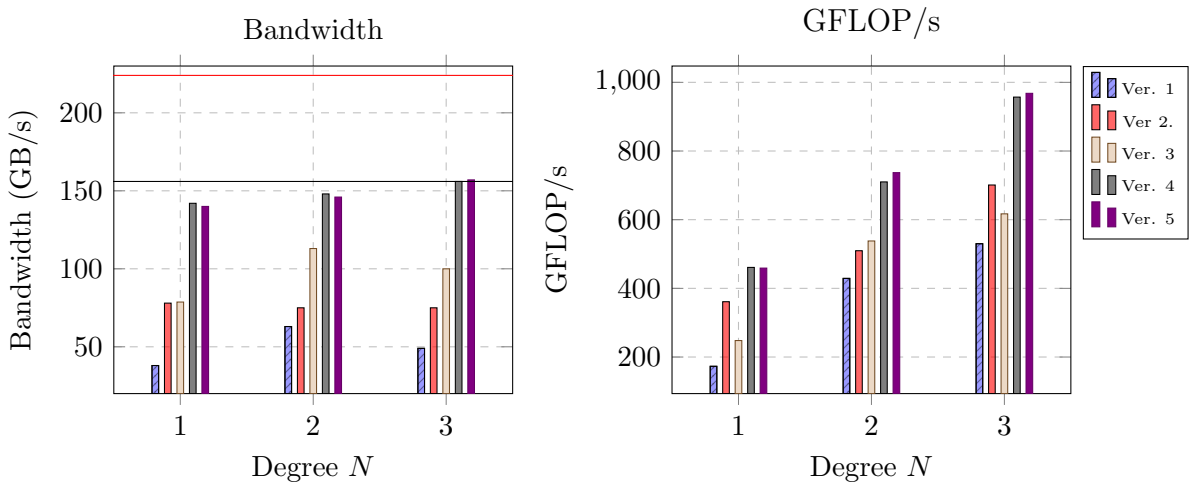


Figure 5.2 : Progressive optimizations of the interpolation kernel. The solid red line corresponds to the theoretical bandwidth as reported by NVIDIA. The black line corresponds to the observed streaming bandwidth as observed by Mei et al. [2]. The observed performance is measured through the NVIDIA profiler.

Table 5.1 : Time to Solution in Sec (Speed up) for the different iterations of Interpolation Kernels.

Grid Size	Order	Ver. 1	Ver. 2	Ver. 3	Ver. 4	Ver. 5
$220 \times 220 \times 220$	$N = 1$	0.095	0.045	0.045	0.024	0.024 (4x)
$150 \times 150 \times 150$	$N = 2$	0.062	0.052	0.035	0.027	0.023 (2.7x)
$110 \times 110 \times 110$	$N = 3$	0.065	0.047	0.041	0.23	0.023 (2.82x)

5.3.2 Hermite Time Stepping Schemes on the GPU

With the polynomial reconstruction procedure described in the previous section I now focus on the implementation of Hermite-Taylor and Hermite-Leapfrog time-stepping. For each reconstructed polynomial the evolution procedure can be performed locally by copying the degrees of freedom of the interpolant onto a rank 3 tensor

$$(\mathbf{Ru})[n_3][n_2][n_1],$$

where n_3, n_2, n_1 range from $0, \cdots, 2N+1$, corresponding to the order of spatial derivative in each spatial dimension. By employing this type of data structure spatial differentiation may be easily carried out as a series of matrix-matrix multiplications using the operator

$$\mathbf{D}_{ij} = \begin{cases} \frac{i+1}{h} & , \quad j = i+1 \\ 0 & , \quad \text{otherwise}, \end{cases} \qquad 0 \leq i, j \leq 2N+2,$$

For convenience $\mathbf{D}_{x_1}$ will correspond to an operator which differentiates with respect to the x_1 dimension and $\mathbf{D}_{x_2}$, $\mathbf{D}_{x_3}$ will correspond to operators which will differentiate with respect to the x_2, and x_3 dimensions. For further clarity Algorithm 4 illustrates applying the differentiation matrix along the x_1 dimension through a series of for loops. Differentiating the reconstructed polynomial in the remaining dimensions is accomplished analogously. Computing higher order derivatives or mixed derivatives may be accomplished by multiplying derivative matrices together. For example the matrix $\mathbf{D}_{x_1}^{(2)} = \mathbf{D}_{x_1}\mathbf{D}_{x_1}$ can be constructed to compute second derivatives along the x_1 dimension.

Algorithm 4 Differentiation in the x_1-dimension

```
1: procedure DifferentiationIn x1(D_x1, Ru, Ru_x)
2:     for tz = 0, 2N + 1 do
3:         for ty = 0, 2N + 1 do
4:             for tx = 0, 2N + 1 do
5:                 if tx < 2N + 1 then
6:                     p_x1 = ((tx+1)/h_x) Ru[tz][ty][tx + 1]
7:                 else
8:                     p_x1 = 0
9:                 Ru_x[tz][ty][tx] = p_x1
```

Hermite-Taylor Evolution

Using the compact notation introduced in the previous section, the Hermite-Taylor algorithm can be expressed as a q-stage loop as listed in Algorithm 5 where q corresponds to the of the Temporal expansion. Similar to the polynomial reconstruction

Algorithm 5 Hermite-Taylor evolution

1: **procedure** HERMITE-TAYLOR($\mathbf{D}_{x_1}$,$\mathbf{D}_{x_2}$,$\mathbf{D}_{x_3}$,$\mathbf{Ru}$)
2: $\quad\hat{\mathbf{w}} = \mathbf{Ru}$
3: $\quad$**for** $k = q, q-1, \ldots, 1$ **do**
4: $\quad\quad\hat{\mathbf{w}} = \mathbf{Ru} + \frac{\Delta t}{k}(\mathbf{D}_{x_1}\hat{\mathbf{w}} + \mathbf{D}_{x_2}\hat{\mathbf{w}} + \mathbf{D}_{x_3}\hat{\mathbf{w}})$
5: $\quad\mathbf{Ru} = \hat{\mathbf{w}}$

kernel, two levels of parallelism are exposed: a coarse level in which each block of threads carries out the Hermite-Taylor scheme for a number of cells and a fine-grained level in which threads carry out the computations for each degree of freedom. Numerical experiments demonstrated that increasing the number of stages, q, in the scheme increases computational intensity. Peak performances were observed when assigning a block of threads to evolve the solution at 4, 2, and 1 cells for orders N=1, 2, and 3 respectively. Peak performance results are reported in Figure 5.3 for varying number of stages. Although the results are profiled with a varying number of stages preserving spatial order of convergence only requires a temporal truncation slightly higher than the rate of convergence of the Hermite interpolant to ensure that the dominating error is caused by the interpolation procedure.

Hermite-Leapfrog Schemes

Unlike the Hermite-Taylor method, the Hermite-Leapfrog scheme is a multi-step method which propagates the solution at two time-steps. For each given node the

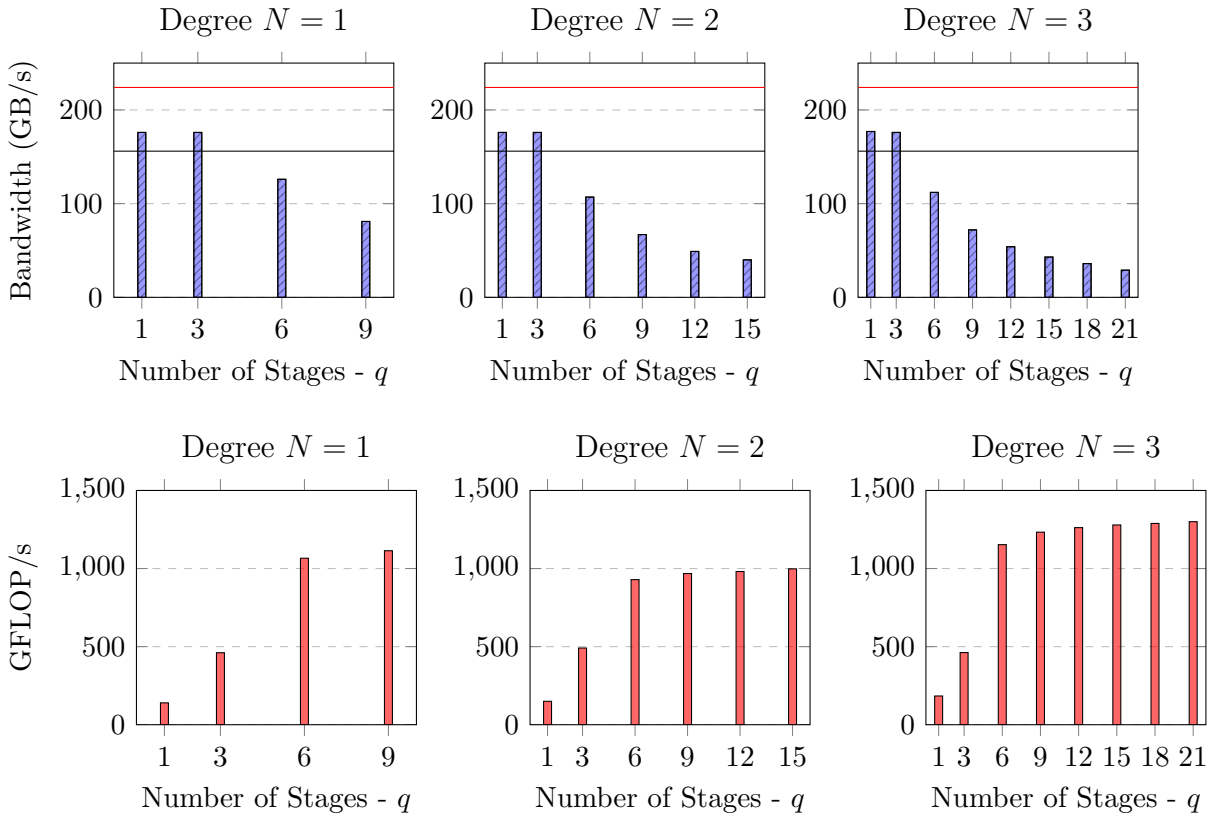


Figure 5.3 : Performance of the Hermite-Taylor kernel, the kernel assigns the evolution at 4, 2, and 1 cells per block of threads for orders $N = 1, 2, 3$ respectively. As the order of the temporal expansion increases the kernel becomes more compute intensive. The solid red line corresponds to the theoretical bandwidth as reported by NVIDIA. The black line corresponds to the observed streaming bandwidth as observed by Mei et al. [2]. The observed performance is measured through the NVIDIA profiler.

Hermite-Leapfrog scheme approximates the solution at t_{k+1} using the following update formula

$$p(\mathbf{x}, t_{k+1}) = -p(\mathbf{x}, t_k) + \sum_{j=0,even} \frac{(\Delta t)^j}{j!} \frac{\partial^j p}{\partial t^j}(\mathbf{x}, t_{k+1/2}).$$

Algorithm 6 illustrates how the summation may be carried out as a $N-1$ stage loop after applying the Cauchy Kowelasky recurrence relation to exchange time derivatives for space derivatives. The Hermite-Leapfrog scheme for the first order acoustic wave equations follows a similar implementation but requires different kernels for the pressure and velocity terms. Peak performance is achieved when a block of threads evolves the solution for multiple cells, peak performance is reported in figures 5.4.

Algorithm 6 Hermite-Leapfrog for the Second Order Wave Equation

1: **procedure** HERMITE-LEAPFROGEVOLUTION($\mathbf{D}^{(2)}_{x_1}$,$\mathbf{D}^{(2)}_{x_2}$,$\mathbf{D}^{(2)}_{x_3}$,**Ru**,**unew**)
2: **unew** $= a_0$**Ru**
3: **for** $k = 1 \cdots N-1$ **do**
4: **Ru** $= 2\frac{(\Delta t)^{2j}}{2j!}\left(\mathbf{D}^{(2)}_{x_1}\mathbf{Ru} + \mathbf{D}^{(2)}_{x_2}\mathbf{Ru} + \mathbf{D}^{(2)}_{x_3}\mathbf{Ru}\right)$
5: **unew**$+=$ **Ru**

5.3.3 Roofline Analysis

As an additional metric to measure performance I consider the roofline model. The roofline model relates flops, bandwidth, and hardware to providing an upper bound on the rate of floating point operations based on the arithmetic intensity of a given kernel [86]. Arithmetic intensity is defined as

$$\text{arithmetic intensity} = \frac{\text{FLOPs performed}}{\text{bytes loaded}}.$$

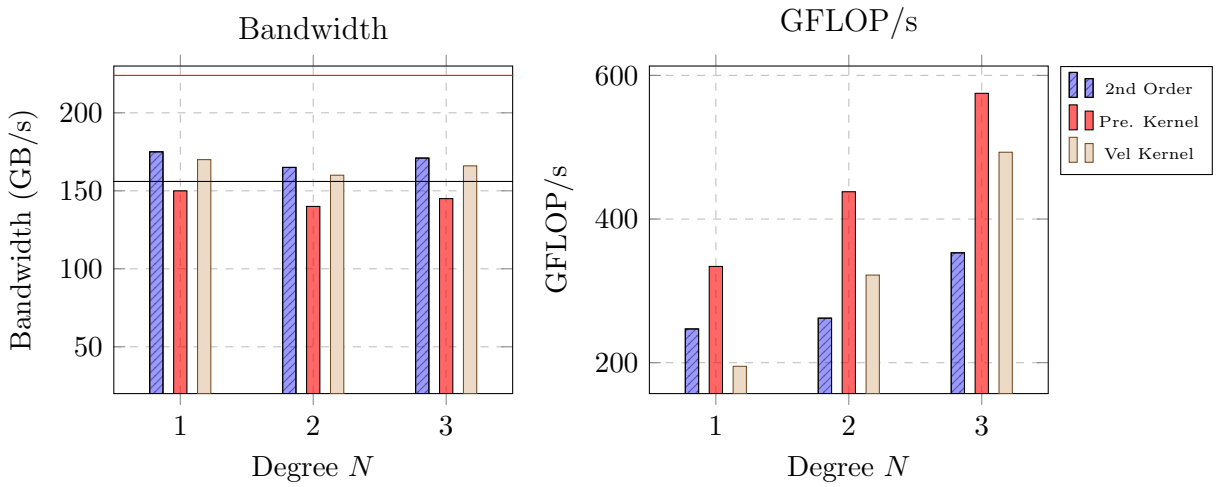


Figure 5.4 : Performance for Hermite-Leapfrog evolution kernel for the second order acoustic wave equation and first order pressure-velocity system. The Hermite-Leapfrog scheme for the pressure-velocity system requires separate kernels for the pressure and velocity. The solid red line corresponds to the theoretical bandwidth as reported by NVIDIA. The black line corresponds to the observed streaming bandwidth as observed by Mei et al. [2]. The observed performance is measured through the NVIDIA profiler.

Pairing the arithmetic intensity with the physical capabilities of the hardware allows the roofine model to present a theoretical ceiling on performance for a given kernel. Theoretical achievable performance is defined as

$$\min(\text{arithmetic intensity} \times \text{peak bandwidth}, \text{peak GFLOP/s}).$$

Figures 5.5 and 5.6 report the performance of the kernels with respect to the roofline model and additionally reports computational efficiency.

Typically there are two types of computational bottle necks, bandwidth or compute. Kernels which are bandwidth limited are constrained by a device's ability to read and write to global memory. Compute bound kernels are limited by the device's ability to perform floating point operations. The roofline model places kernels limited

by bandwidth on the bottom left while compute bound kernels are found on the top right.

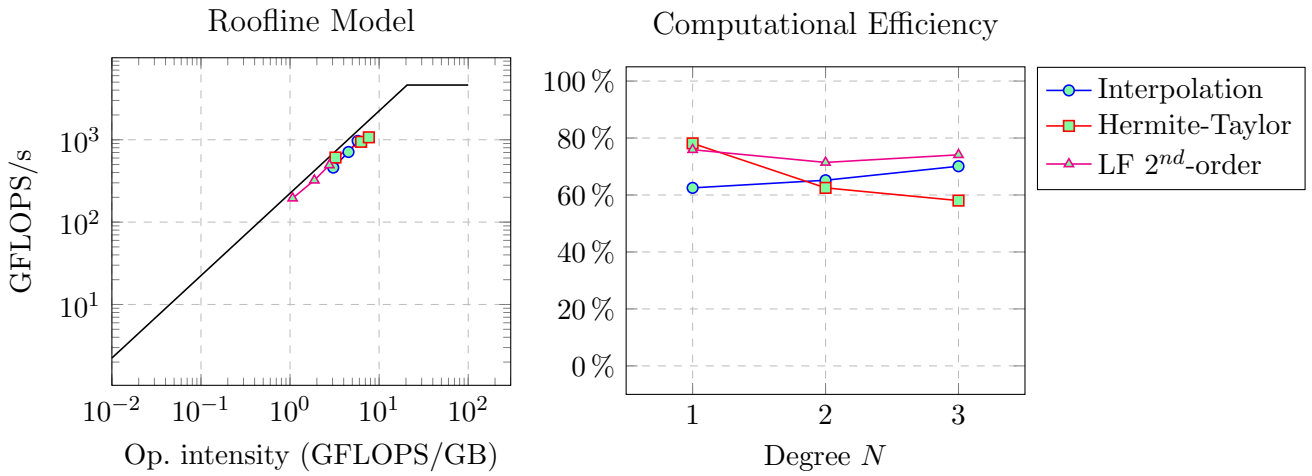


Figure 5.5 : Roofline performance analysis for the Hermite Interpolation, Hermite-Taylor, and Hermite-Leapfrog evolution kernels.

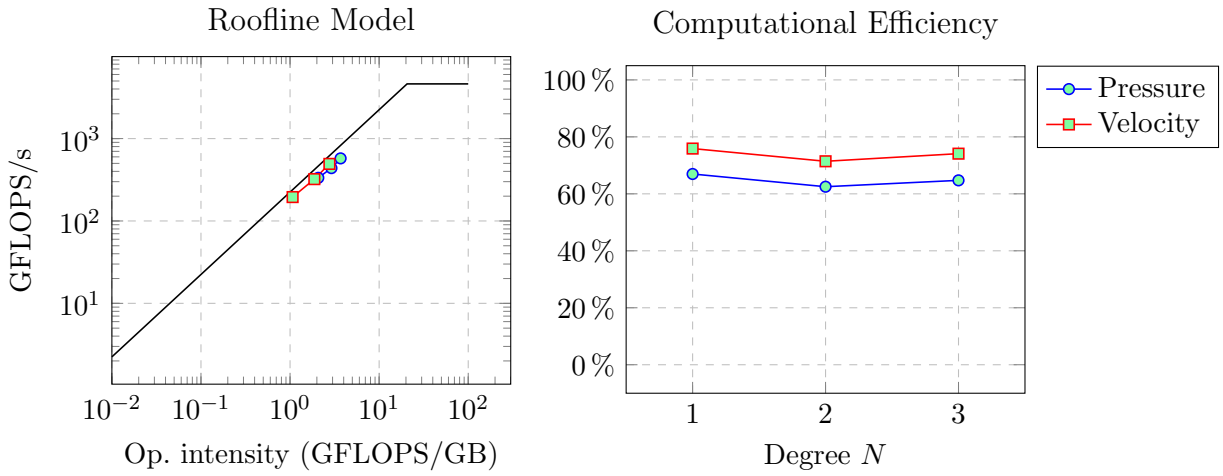


Figure 5.6 : Roofline and performance analysis for the Hermite-Leapfrog evolution kernels for the pressure and velocity fields. The Hermite-Taylor kernels are profiled using a $q = 2N + 2$ stage loop.

5.3.4 A Monolithic Kernel

A two kernel approach allows for fine tuning of each individual procedure at the cost of storing the coefficients for the reconstructed polynomial. In the interest of minimizing the amount of needed global memory, the interpolation and evolution step were fused into a single monolithic kernel. The monolithic kernel removes the need to store the interpolant by carrying out the interpolation and evolution step in a single kernel call. The caveat of this implementation is that it does require more shared memory compared to two splitting the interpolation and evolution step.

For completeness numerical experiments are carried out using the Hermite-Taylor scheme and the Leapfrog-scheme for the first and second order wave equation respectively. Figure 5.7 reports the observed bandwidth and GFLOP/s of the monolithic kernels. These kernels exchange high bandwidth performance for a high flop performance. Tables 5.2 and 5.3 compare time to solution and demonstrate that fusing the kernels provides a comparable time to solution to splitting the kernels. The major advantage of these kernels is the reduction of global memory needed which is beneficial given that GPUs typically offer less global memory than a CPU + RAM configuration.

	$N = 1$	$N = 2$	$N = 3$
Hermite-Taylor: Monolithic Kernels	2.01 (sec)	13.18 (sec)	40.00 (sec)
Hermite-Taylor: Two Kernels	3.06 (sec)	14.54 (sec)	39.30 (sec)

Table 5.2 : Comparison of time to solution. The initial conditions were propagated for 200 time-steps on a fixed grid of 70 grid points in each dimension.

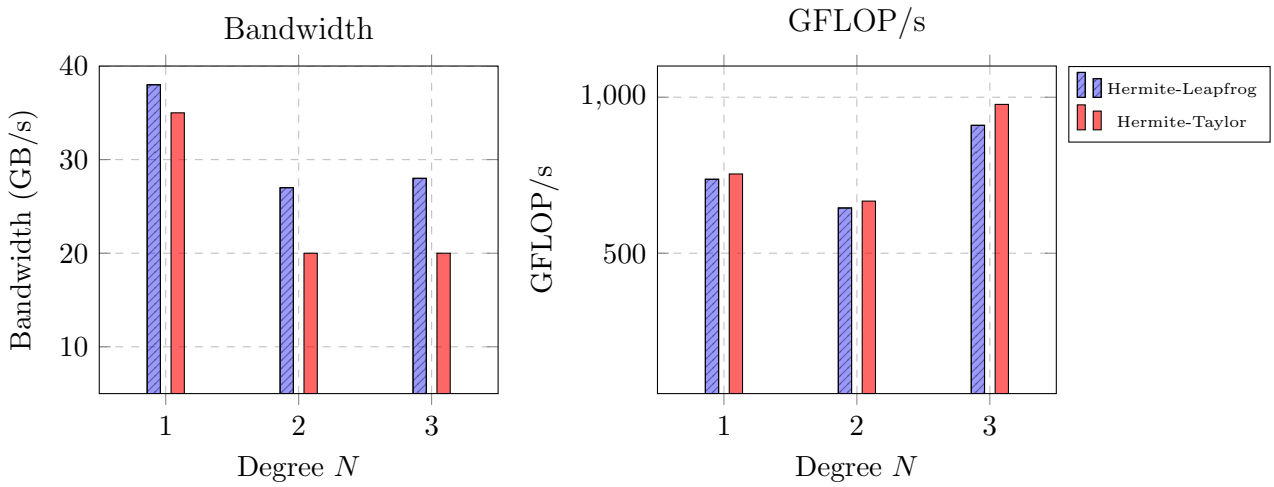


Figure 5.7 : Performance for the Hermite-Leapfrog monolithic kernel tailored for the second order acoustic wave equation. The observed performance is measured through the NVIDIA profiler.

	$N = 1$	$N = 2$	$N = 3$
Hermite-Leapfrog: - Single Kernel	1.32 (sec)	7.25 (sec)	19.61 (sec)
Hermite-Leapfrog: - Two Kernels	2.45 (sec)	8.96 (sec)	19.50 (sec)

Table 5.3 : Comparison of time to solution. The initial conditions were propagated for 200 time-steps on a fixed grid of 110 grid points in each dimension.

5.4 Optimizing Across Devices

OCCA provides a platform in which a developer may cross-compile their code into a variety of API's enabling cross-platform portability. Of course specific hardware tuning is left to the developer and considerations must be made when targeting specific devices. For example, there is no guarantee that a well optimized OCCA kernel for the GPU will perform exceptionally well on the CPU and vise versa. Furthermore, kernels which are designed to be truly portable must abide to the CUDA or OpenCL programming model in which loops are held to a fixed sized, the fixed loops sizes corresponds to the predefined compute grid specified for a GPU kernel.

This section discusses modest optimization that were carried out for the Hermite-Leapfrog scheme tailored to the second order wave equation when targeting the CPU. Building from the GPU tuned monolithic kernel, numerical experiments demonstrated (Table 5.4) that additional performance may be extracted by varying loop sizes and removing branching (which may be unavoidable in a GPU implementation). For

Grid Size	Order	GPU Tailored	CPU Tailored
220x220x220	$N = 1$	4.28 (sec)	1.78 (sec)
150x150x150	$N = 2$	7.22 (sec)	2.28 (sec)
110x110x110	$N = 3$	7.86 (sec)	2.46 (sec)

Table 5.4 : Time to solution comparing GPU and CPU tailored Hermite-Leapfrog monolithic kernels on the CPU. Numerical experiments were run on an Intel i7-4790k with 4.00GHz using 8 threads.

completeness, a comparison of time to solution is presented comparing a CPU and GPU tailored kernels. A comparison of time to solution is presented by propagating the solution for 200 time steps on a fixed grid of 150 grid points in each dimension. Table 5.5 reports the observed run-time of the kernels without accounting for memory transfers. Noticeably the GPU implementation provides roughly a 12-14x speed up compared to the multi-core CPU version (Intel i7-4790k with 4.00GHz) using 8 threads.

	$N = 1$	$N = 2$	$N = 3$
OCCA::OpenMP	25.04 (sec)	100.04 (sec)	266.32 (sec)
OCCA::CUDA - Single Kernel	1.32 (sec)	7.25 (sec)	19.61 (sec)
OCCA::CUDA - Two Kernels	2.45 (sec)	8.96 (sec)	19.50 (sec)

Table 5.5 : Comparison of time to solution of Hermite kernels executed on the GPU and CPU. The initial condition is propagated for 200 time-steps on a fixed grid of 110 grid points in each dimension. The time solution compares kernel run time on an NVIDIA GTX 980 and an Intel i7-4790k with 4.00GHz using 8 threads.

5.5 Multi-GPU implementation

The limited amount of global memory found on the NVIDIA GTX 980 (4GB) quickly places a ceiling on the size of the problems that may be considered on a single GPU. For example a typical computational unit of the seismic imaging algorithm "Reverse Time Migration" easily exceeds 10GB of memory [1, 76]. By employing multiple GPUs the workload may be distributed across multiple devices. For clarity I describe domain decomposition using two GPUs which may easily be generalized to more GPUs. The multi-GPU implementation is carried out using the Dual Hermite method as it is of the more intricate methods and periodic boundary conditions are assumed.

With two GPUs the domain is partitioned with an equal number of primary nodes on each processor. Figure 5.8 illustrates the partitioning of a two-dimensional periodic domain. Transferring data between different devices is handled by standard MPI calls. As each GPU maintains half of the data it is necessary to supplement

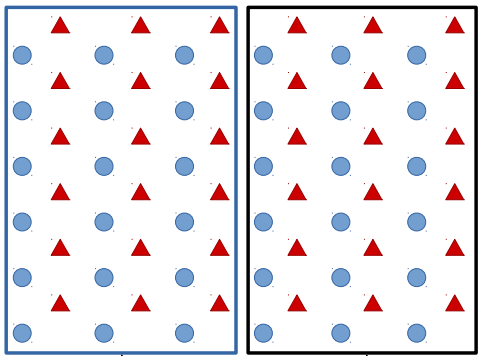
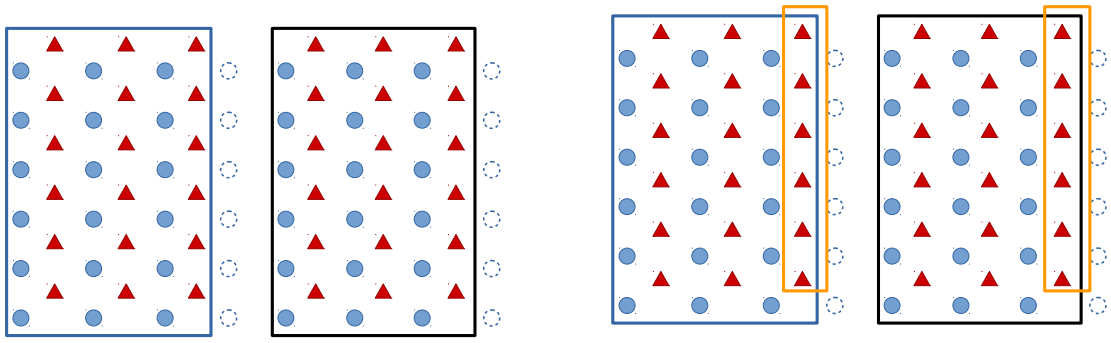


Figure 5.8 : Example of decomposing the computational domain across two processes.

each partitioned domain with ghost nodes. The role of ghost nodes is to provide supplementary data necessary for the interpolation procedure. Figure 5.9 illustrates the introduction of ghost nodes into the domain. The ghost nodes are essential for constructing the interpolant on the right edge of the domain. The computation is

then further divided into two stages. The first stage performs the interpolation on the highlighted dual nodes as illustrated in Figure 5.9. In order to maximize scalability the computation of the remaining domain is overlapped with the exchange in ghost nodes. As MPI is used to exchange ghost nodes this requires three memory copies. The ghost nodes are transferred from the GPU to the CPU and exchanged between CPU processes. Lastly the data is copied from the CPU back onto the GPU. Enabling the GPU and CPU to continue processing during the ghost node exchange is CUDA's ability to perform asynchronous kernel execution and memory copy calls. Table 5.6 illustrates the strong scaling results on up to three NVIDIA GTX 980 GPUs.



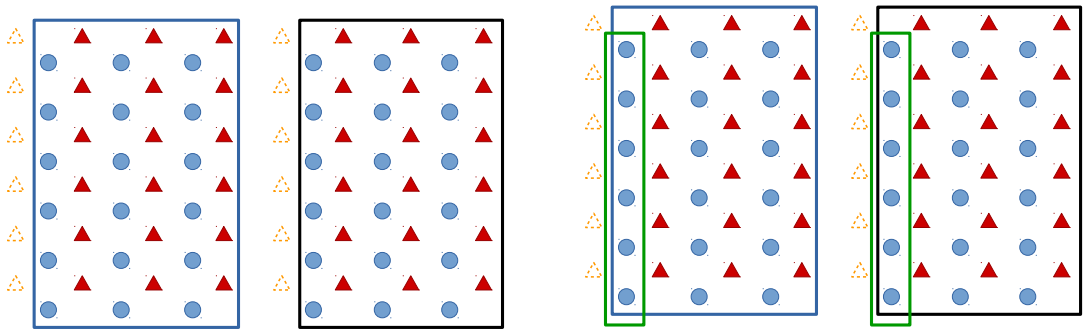
1. Primal Ghost Nodes

2. Update Dual Nodes

Figure 5.9 : Stage 1 of domain decomposition. Ghost nodes are introduced on the primary grid (blue circles) to supplement the domain with the necessary degrees of freedom to perform the Hermite interpolation.

Table 5.6 : Strong scaling on three GPUs

Order and Grid Dimension	1 GPU	2 GPU	3 GPU
$N = 1$: 160x160x480	1.00	2.2	3.32
$N = 2$: 100x100x300	1.00	1.97	2.91
$N = 3$: 80x80x240	1.00	2.05	2.98



3. Dual Ghost Nodes 4. Update Primal Nodes

Figure 5.10 : Stage 2 of domain decomposition. Ghost nodes are introduced on the dual (red triangles) grid to supplement the domain with the necessary degrees of freedom to perform the Hermite interpolation.

5.6 Summary

This chapter described the implementation details and performance analysis for a three-dimensional GPU accelerated Hermite method solver using Hermite-Taylor and Hermite-Leapfrog time-stepping. The pioneering work in tailoring Hermite methods was first carried out by Dye in his masters thesis wherein he described the implementation details for a two dimensional advection equation [16].

In this work a two kernel design is first considered which allows for fine tuning of the interpolation and evolution procedure. Second a single monolithic kernel is studied in which the interpolation and evolution steps are fused together into a single kernel. The advantage of a monolithic kernel is the reduction of global memory needed as the interpolant is not explicitly stored in global memory furthermore the monolithic kernel provided a comparable time to solution to using two kernel.

As CPUs also provide a platform for parallelism a discussion on tailoring Hermite methods to multi-core CPUs is provided along with a to time to solution comparison to their GPU counter parts. It is observed that truly hardware portable code

(within the OCCA framework) requires fixing loop sizes to abide with the most restrictive programming model in this case the CUDA/OpenCL programming model. By exchanging hardware independence for performance, computational experiments demonstrated performance on the CPU may be improved by varying loop sizes. Lastly to enable large scale simulations a distributed memory implementation is presented using an the MPI + GPU programming model along with strong scaling results.

Chapter 6

Hermite Methods for Wave Propagation

In this chapter I tie together the newly developed Hermite methods and their high-performance implementations to simulate wave propagation in various contexts. The starting point of this chapter is the simulation of wave propagation as governed by linear hyperbolic equations. Numerical experiments are presented using various equations to illustrate the efficiency and performance of the methods.

As simulating wave propagation in a large unbounded domain is of much interest for applications, I discuss the implementation of Bérenger's perfectly matched layers (PML) for Hermite methods. To guide parameter choices for a Hermite-PML framework a the results of a series of one dimensional numerical experiments are presented. The Hermite-PML framework is then used to simulate wave propagation on a synthetic velocity model. Lastly, I present an overview of techniques for evaluating nonlinearities with Hermite methods and present simulations of nonlinear wave propagation.

6.1 Simulating Linear Wave Propagation

Chapters 3, and 4 introduced variations of the Dual Hermite scheme (Chapter 2) and with each new method accuracy and performance was assessed with known analytic solutions. In this section I begin to expand on the types of equations considered and carry out numerical experiments with equations where the solution is unknown.

6.1.1 Elastic Wave Equations

As the first set of numerical experiments, I consider the elastic wave equations. These equations have found applications as high fidelity models for simulating wave propagation through the earth's subsurface, [87, 88, 89]. The elastic wave equation may be expressed as a system of second order equations

$$\rho \frac{\partial^2 \mathbf{u}}{\partial t^2} = \nabla \cdot \mathcal{T} + \mathbf{f}, \quad \mathbf{x} \in \Omega, t \geq 0 \tag{6.1}$$

$$\mathcal{T} = \lambda(\nabla \cdot \mathbf{u})\mathbf{I} + \mu\left(\nabla \mathbf{u} + \nabla \mathbf{u}^T\right). \tag{6.2}$$

Here $\mathcal{T}$ is referred to as the stress tensor, the vector $\mathbf{u} = (u, v, w)^T$ corresponds to the displacement vector in Cartesian coordinates, and $\mathbf{f}$ is the external forcing. The density of the material is characterized by $\rho(\mathbf{x}) > 0$, while $\lambda(\mathbf{x}) > 0$ and $\mu(\mathbf{x})$ denote the Lamé parameters which describe the elastic and shear modulus. When $\mu = 0$ these equations simplify to the second order acoustic wave equation. As these equations have second order time-derivatives, the Hermite-Leapfrog scheme is used to propagate the solution. These equations are discretized in the domain $[0, 1]^3$ and a standing wave solution is used to validate the implementation.

Standing Wave Solution

The standing wave solutions for the elastic wave equations are given by

$$\begin{aligned} u(\mathbf{x}, t) &= \cos(\omega t)\sin(\omega x)\sin(\omega y)\sin(\omega z) \\ v(\mathbf{x}, t) &= \cos(\omega t)\sin(\omega x)\cos(\omega y)\sin(\omega z) \\ w(\mathbf{x}, t) &= \cos(\omega t)\sin(\omega x)\sin(\omega y)\cos(\omega z), \end{aligned}$$

and the forcing function, $\mathbf{f}$, is chosen to enforce the standing wave solution. For these numerical experiments ω is chosen to be 2π. The solution is propagated to a final time of $T = 2.0$. Table 6.1 reports the observed accuracy and rates of convergence. The observed rates, $O(h^{2N})$ remain consistent with the numerical experiments in Chapter 4.

	$N = 1$		$N = 2$		$N = 3$	
h	L^2 Error	rate	L^2 Error	rate	L^2 Error	rate
0.1	0.0028598	-	1.52496e-05	-	7.51775e-09	-
0.05	0.00483659	-	9.5524e-07	3.99	1.17237e-10	6.00
0.025	0.00148922	1.69	5.98616e-08	3.99	1.83457e-12	5.99
0.0125	0.000378226	1.97	-	-	-	

Table 6.1 : Observed accuracy and convergence rates for the three-dimensional elastic wave equation under Hermite-Leapfrog time-stepping using an N order method. These equations were solved as a system in second order form using Hermite-Leapfrog time stepping.

Propagation of Primary and Secondary Waves

As a second set of numerical experiments, I consider wave propagation stemming from a Gaussian initial condition. This initial condition serves as an approximation of a point source and under this type of initial condition the elastic wave equations exhibit two types of propagating waves. The first is the primary wave (or pressure wave) which is a result of the change of compressions and rarefactions. The secondary wave (or shear wave) is caused by shear effects and follows the primary wave but moves at a slower velocity. The initial condition here is placed on the u component of the system

$$u(\mathbf{x}, 0) = 0, \quad u(\mathbf{x}, t + \Delta t/2) = \exp\left(-\frac{\|x\|^2}{0.002}\right).$$

For these numerical experiments, the material constants are assumed to be equal to one. Figures 6.1 illustrate the propagation of the wave across the u displacement component.

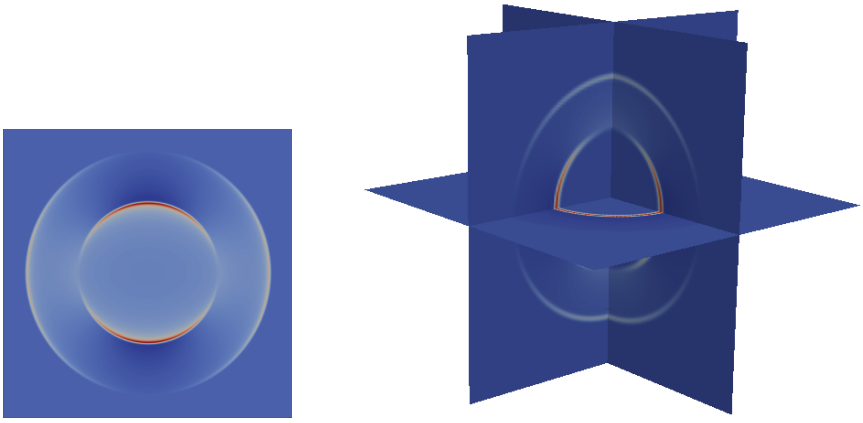


Figure 6.1 : Illustration of the primary and secondary wave propagating via the elastic wave equations. The discretization was carried out using an $N = 2$ Hermite-Leapfrog method on the domain $[-1, 1]^3$.

6.1.2 Acoustics in Discontinuous Media

As a third set of numerical experiments I turn to the acoustic wave equations and consider wave propagation over discontinuous media. In these numerical experiments, the computational domain is chosen to be $[0, 1]^3$ and material property is designed with a discontinuous transition between two velocities at $z = 0.5$. The top half of the domain propagates waves at unit wave speed while the bottom half propagates waves at twice the speed. Although an analytic solution is not known, a reflection is expected as the wave transition between velocities. Figures 6.2 illustrates the discontinuous velocity model in which the wave is propagated on and a snapshot of the numerical experiment is found on the right. It can be observed that Hermite methods are able to capture the reflection.

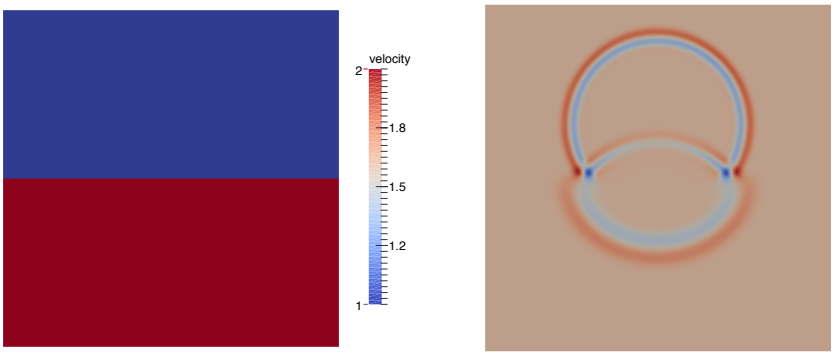


Figure 6.2 : (Left) Velocity model with a discontinuous transition. (Right) The result of using an order $N = 1$ Hermite method to propagate the wave. A reflection is observed as the wave transitions between velocities. The computational domain is chosen to be $[0, 1]^3$ with velocity transition at $z = 0.5$.

6.2 Absorbing Boundary Layers

An important aspect in modeling wave propagation is the ability to truncate unbounded domains in order to model wave propagation in a region of interest. The experiments carried out in Chapters 3, 4, and Section 6.1 all assumed periodic boundary conditions. In this section, I discuss the development of simulating wave propagation in a truncated geometry using the Hermite method framework for the acoustic wave equations.

Simulating wave propagation in a truncated domain is typically accomplished by the use of absorbing boundary conditions [90, 91] or dampening layers [92, 93, 94, 41, 95]. A popular boundary layer is Jean-Pierre Bérenger's perfectly matched layers (PML). This formulation was first introduced as a technique for solving Maxwell's equations in an unbounded medium [96]. Since its introduction, Bérenger's method has been reformulated and generalized for many wave-like problems [97, 98, 99].

Introducing PML to a wave system consist of adding a system of auxiliary equations and dampening terms. The key ideas behind the PML is to introduce a "frame"

into the domain in which the waves exponentially decay. Boundary conditions may be taken as any, however, the global performance of the absorbing layers may be further improved by paring the PML with absorbing boundary conditions [100].

To introduce PML to the acoustic wave equations, I consider the split-field formulation as described in [101]. In this formulation, the pressure field is decomposed into each Cartesian coordinate, $p = p_x + p_y + p_z$, and the dampening functions $\sigma_x(x)$, $\sigma_y(y)$, and $\sigma_z(z)$ are introduced for each coordinate. The acoustic wave equations within the PML framework is then formulated as

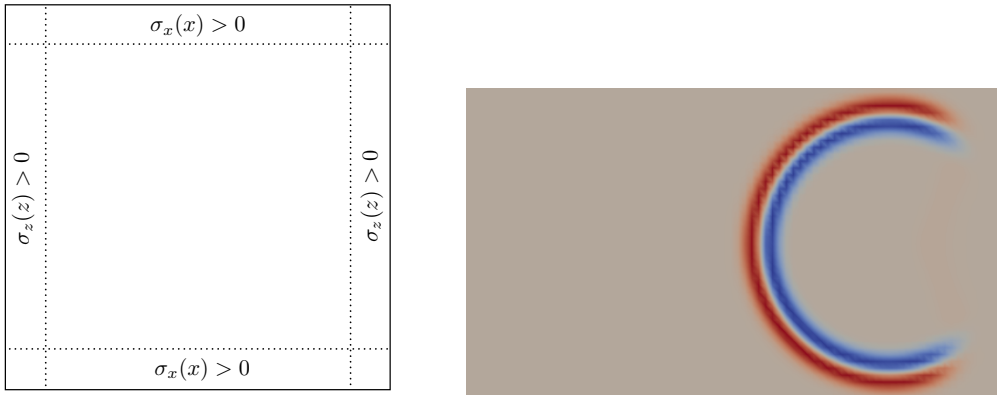
$$\begin{aligned}
\frac{\partial p_x}{\partial t} + \frac{\partial v_x}{\partial x} &= -\sigma_x p_x, \qquad & \frac{\partial v_x}{\partial t} + \frac{\partial p}{\partial x} &= -\sigma_x v_x \\
\frac{\partial p_y}{\partial t} + \frac{\partial v_y}{\partial y} &= -\sigma_y p_y, \qquad & \frac{\partial v_y}{\partial t} + \frac{\partial p}{\partial y} &= -\sigma_y v_y \\
\frac{\partial p_z}{\partial t} + \frac{\partial v_z}{\partial z} &= -\sigma_z p_z, \qquad & \frac{\partial v_y}{\partial t} + \frac{\partial p}{\partial z} &= -\sigma_y v_y.
\end{aligned} \tag{6.3}$$

Figure 6.3 illustrates the computational domain and the PML "frame". The absorbing functions take a positive value within the frame causing the wave to decay. Thus choices in the thickness and absorption coefficients will affect the overall quality of the absorption.

6.2.1 Choices in Absorption Functions

The role of the absorption function is to gradually introduce the PML. These functions are designed to be zero within the computational domain and gradually increase in value towards the edge of the domain; too sharp of a transition will introduce reflections into the simulation. The most commonly used absorption functions are polynomials of the form

$$\sigma_n(x) = \alpha \left(\frac{x}{\delta}\right)^n, \tag{6.4}$$



(a) 2D PML (b) Wave being absorbed.

Figure 6.3 : Illustration of PML added to a computational domain (Left) and a wave absorbed by the PML (Right).

where n corresponds to the order of the polynomial. The parameter α is used to adjust the maximum value of the coefficient and δ is used to set the thickness of the layer. The most popular absorption function is the parabolic function, $n = 2$. As an alternative to polynomial functions, the work of Bermúdez et al. suggests the use of hyperbolic functions

$$\sigma_h(x) = \frac{\alpha}{\delta - x}. \tag{6.5}$$

In comparison to polynomial functions, hyperbolic functions have been demonstrated to be more effective than polynomials for finite-element formulations [101, 102]. In order to guide the choice in absorption functions for Hermite methods, a series of one-dimensional experiments are carried out using varying orders of polynomial and hyperbolic functions.

6.2.2 One-Dimensional Benchmark

To guide the choice in absorption functions within the Hermite method framework, I carry out the one-dimensional benchmarks as proposed by Modave in [101]. The one-dimensional pressure-velocity system with PML is given by

$$\frac{\partial p}{\partial t} + \frac{\partial v}{\partial x} = -\sigma p \tag{6.6}$$

$$\frac{\partial v}{\partial t} + \frac{\partial p}{\partial x} = -\sigma v. \tag{6.7}$$

For these numerical experiments, the computational domain is chosen to be $[-L, 0]$ and extended with a finite PML region $\Sigma = [0, \delta]$. An incident Gaussian pulse is used as an initial condition

$$p(x, 0) = \exp\left(-\frac{(x + L/4)^2}{R^2}\right) \tag{6.8}$$

$$v(x, 0) = \exp\left(-\frac{(x + L/4)^2}{R^2}\right). \tag{6.9}$$

As boundary conditions I impose zero Dirichlet boundary conditions on the pressure term. Differentiating the boundary condition in time and utilizing the equation leads to the following statements

$$p = 0, \quad x = x_L, x_R \tag{6.10}$$

$$\frac{\partial v}{\partial x} = 0, \quad x = x_L, x_R, \tag{6.11}$$

where x_L and x_R are the left and right end points of the computational domain. These boundary conditions are implemented within the Hermite framework by mirroring the polynomials on the boundaries so that the reflected polynomial is either odd or even

thus corresponding to the Dirichlet or Neumann boundary conditions respectively. The reflection of the PML is quantified by computing the relative error ξ_r as defined by

$$\xi_r = \sqrt{\frac{E_{pml}(T_f)}{E_{wall}(T_f)}}. \tag{6.12}$$

The term E_{pml} is the total energy of the numerical solution with the PML at the final time. The total energy is computed as

$$E_{pml}(t_f) = \int_{-L}^{0} \left(\frac{1}{2} p^2(x, t_f) + \frac{1}{2} u^2(x, t_f) \right) dx.$$

Similarly, E_{wall} corresponds to the numerical solution with the Dirichlet/Nuemman boundary conditions.

The one-dimensional numerical experiments are carried out in a domain of length $L = 50$ with a PML thickness of $\sigma = 5$. The solution is propagated to a final time of $T = 55$. Figure 6.4 reports the relative error ξ_r as a function of α (the magnitude of the PML intensity). Although numerical experiments demonstrate that increasing the α parameter greatly increases the quality of the PML, instabilities were observed when the value is sufficiently large. The point of instability is dependent on the order of the scheme and choice in coefficients. Furthermore, in order to avoid the singularity with the hyperbolic absorption function, the thickness is adjusted to be $\sigma + \epsilon$, where ϵ is chosen to be small.

As a general guideline, numerical experiments suggest that a quadratic function can provide a comparable quality of absorption when compared to the hyperbolic function. From an implementation standpoint using a quadratic function is also advantageous as high order derivatives vanish reducing the amount of computation

and storage.

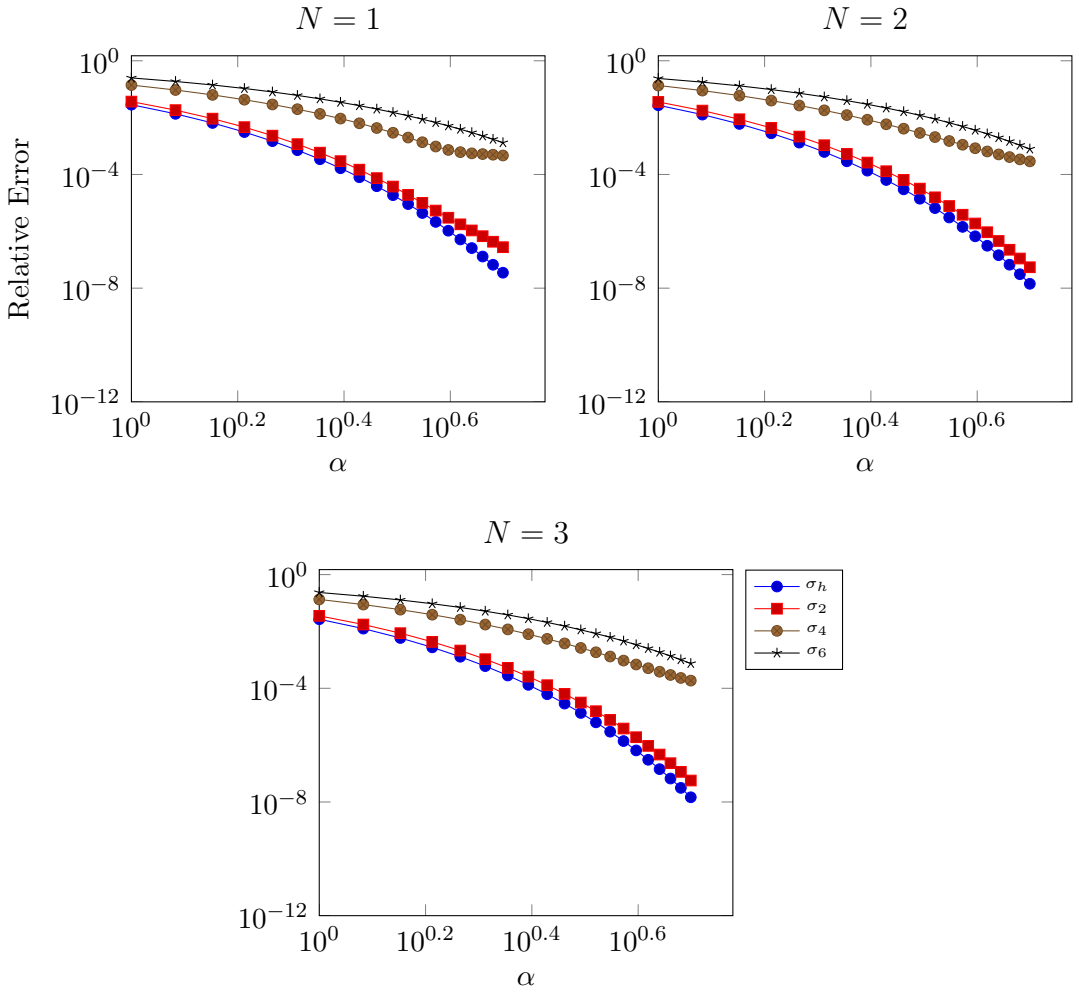


Figure 6.4 : One-dimensional benchmarks of perfectly matched layers using an N^{th} order Dual Hermite method with various absorption functions. The coefficient σ_h corresponds to the hyperbolic function while σ_n corresponds to an n^{th} order polynomial. Relative errors are reported as a function of the absorption parameter α.

.

6.3 Wave Propagation in Smooth Media

In consideration for the development of tools for simulating wave propagation on heterogeneous media; I present a set of numerical experiments in which I propagate acoustic waves over a synthetic model with PML. A coarse version of the velocity model "SEG C3WA" is used as a benchmark problem. The velocity model is available from the Society of Exploration Geophysicist website (wiki.seg.org). As the original model is initially discontinuous, the model is smoothed using a Gaussian filter. Spatial derivatives are then computed using sixth order finite difference stencils allowing the velocity model to be expressed as a collection of local Taylor series polynomials.

To initiate wave propagation, the forcing function is assumed to be a product of the Ricker wavelet and a Gaussian function. Figures 6.5 and 6.6 illustrate the wave propagating across the velocity model.

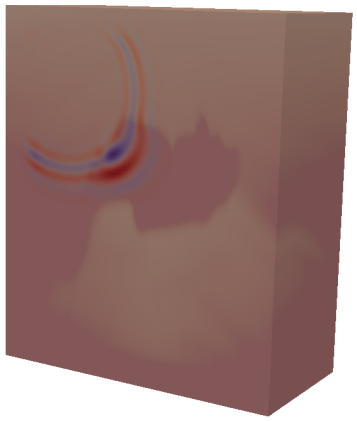


Figure 6.5 : Acoustic wave propagating on a coarse velocity model ("SEG C3WA" available from wiki.seg.org). The model was first smoothed in order in order estimate derivatives. The derivatives enabled the model to be expressed as a collection of Taylor-series expansions. An $N = 1$ order Hermite Runge-Kutta scheme was used to propagate the solution.

Figure 6.6 : Snapshots of an acoustic wave propagating on a coarse velocity model ("SEG C3WA" available from wiki.seg.org). The model was first smoothed in order in order estimate derivatives. The derivatives enabled the model to be expressed as a collection of Taylor-series expansions. An $N = 1$ order Hermite Runge-Kutta scheme was used to propagate the solution.

6.4 Techniques for Evaluating Nonlinearities

Hermite methods can also be used for the numerical solution of nonlinear hyperbolic equations. In the interest of efficiency however, it is advantageous to swap out the Hermite-Taylor/Leapfrog scheme for a one-step ODE integration scheme [22, 52]. The added complexity of using Hermite-Taylor schemes stems from the need to compute high order time derivatives and carrying out polynomial multiplication per right hand side evaluation. For example when considering burgers equation

$$u_t = -uu_x + \nu u_{xx}, \tag{6.13}$$

computing a single time derivative requires polynomial multiplication between u and u_x. The advantage of using a single step method is that only one time derivative of the polynomial needs to be computed. Despite needing only a single time derivative, high dimensional polynomial multiplication has a high computational intensity motivating the need for approaches to reduce the computational burden. For completeness I

provide an overview approaches proposed in the literature.

6.4.1 Polynomial Multiplications via Convolutions

One approach to carryout polynomial multiplication is by convolutions. A convolution in one dimension multiplies two polynomials of degree $2N + 1$ of a single variable, $a(x) = \sum_{i=0}^{2N+1} a_i x_i$ and $b(x) = \sum_{j=0}^{2N+1} b_j x_j$, by carrying out the following operation

$$c(x) = \sum_{k=0}^{4N+2} c_k x_k. \tag{6.14}$$

Here the coefficients, c_k, are computed as $c_k = \sum_{i=0} a_i b_{k-1}$ for $0 \leq k \leq 4N + 2$. The algorithmic complexity of a convolution is given by $O(N^{2d})$, where d corresponds to the dimension of the polynomial. As a method to reduce the computational burden of performing convolutions, Hagstrom and Appelö proposed the use of the fast Fourier transform (FFT) in [22]. The FFT reduces the computational complexity of convolutions from $O(N^{2d})$ to $O(n^d log^d(n))$ by recasting the operations as point-wise multiplications in the frequency domain.

6.4.2 Polynomial Multiplications via Chebyshev Projections

As an alternative to the FFT, Chang Young Jang proposed an alternative method in which the local Taylor polynomials are evaluated over a collection of Chebyshev nodes and the nonlinearities are carried out as point wise operations [**?**]. Similar to the FFT approach of Hagstrom and Appelo, this enables the operations to be carried out as a series of point-wise operations. For clarity I provide an example of this procedure. Without loss of generality I consider a Hermite interpolant over the bi-unit domain.

Chebyshev nodes are generated over the bi-unit domain by

$$x_k = \cos\left((2k-1)/(2N_c)\pi\right), \quad k \in \{1, \ldots, N_c\}. \tag{6.15}$$

As polynomial multiplication of two $2N+1$ polynomials results in a polynomial of order $4N+2$, the polynomials are evaluated over a collection of $N_c = 4N+3$ Chebyshev nodes. Polynomial multiplication may then be carried out as a series of point-wise operations and then projected back to a $4N+2$ polynomial.

With regards to accuracy, Jang demonstrates in his doctoral thesis [59] that this approach preserves the expected rates of convergence of Hermite methods. Additionally, his results demonstrate that this approach is more efficient than both convolution and FFT-based polynomial multiplication in high dimensions.

The approach proposed by Jang has additional utility beyond polynomial multiplication. First, this approach may also be used to incorporate a variety of functions for initial conditions as only function values are needed. Second, this approach is easily extended to evaluate non-multiplicative nonlinearities as I will demonstrate in the next section.

6.5 Simulating Nonlinear Wave Propagation

At the time of writing, Hermite methods have only been applied to nonlinear equations with multiplicative nonlinearities. As a first example I use a system of one-dimensional hyperbolic equations typically used for blood flow modeling. These equations are of particular interest as they have non multiplicative nonlinearities. As a second example I employ Euler's equations for gas dynamics. In both numerical examples I carryout time-stepping using a fourth order Runge-Kutta scheme and adjust the step size in

order to prevent the temporal errors from dominating.

6.5.1 One-Dimensional Blood Flow Equations

In one-dimension, the (A, Q) system as derived by Sunčica Čanić, and Eun Kim [3] are used to model flow through a vessel cross-sectional area (A) and average momentum (Q). The system is expressed as

$$\frac{\partial A}{\partial t} + \frac{\partial Q}{\partial x} = 0, \tag{6.16a}$$

$$\frac{\partial Q}{\partial t} + \frac{\partial}{\partial x}\left(\alpha \frac{Q^2}{A}\right) + \frac{A}{\rho}\frac{\partial A}{\partial x} = -2\pi\nu \frac{\alpha}{\alpha - 1}\frac{Q}{A}. \tag{6.16b}$$

Here the parameter α is assumed to be a constant and referred to as the Coriolis coefficient. The parameters ν and ρ corresponds to the fluid's kinematic viscosity and density. Further details of these equations may be found in [3, 103]. Equations 6.16 are used to exemplify the strengths of Jang's Chebyshev projection technique. Indeed, differentiating terms such as

$$\frac{\partial}{\partial x}\left(\alpha \frac{Q^2}{A}\right), \quad \frac{Q}{A}$$

would require the quotient rule spawning many new terms. To simplify the computation the local polynomials approximating A and Q are first evaluated on Chebyshev nodes and polynomial division is carried out as point-wise operations. The resulting solution is then projected back to a local Taylor-polynomial. In order to assess the accuracy of this approach the method of manufactured solutions is employed. For these numerical experiments the analytic solution is chosen as

$$A(x,t) = \sin(t)\cos(\omega x) + 2, \quad Q(x,t) = \sin(t)\cos(\omega x).$$

	$N=1$		$N=2$		$N=3$	
h	L^2 Error	rate	L^2 Error	rate	L^2 Error	rate
0.2	2.14e-02	-	2.57e-04		9.11e-06	-
0.1	3.47e-03	2.62	8.45e-06	4.92	5.11e-08	7.42
0.05	4.59e-04	2.91	2.85e-07	4.88	4.21e-10	6.92
0.025	5.84-05	2.97	8.77e-09	5.02	3.28e-12	7.00

Table 6.2 : Observed L^2 errors and rates of convergence when using an N order Hermite Runge-Kutta method to solve the (A,Q) blood flow equations as derived [3].

The parameters are chosen to be $\alpha = 1.1$, $\nu = 1$ and $\rho = 1$. The solution is discretized on the bi-unit domain and the solution is propagated to a final time of $T = 2.5$. Table 6.2 illustrates that the rates of convergence may be preserved when using the Jang's Chebychev projection technique [59].

6.5.2 Compressible Flows

In the second set of experiments Euler's equations for compressible flows are considered. In two dimensions the conservative formulation is expressed as

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} + \frac{\partial \rho v}{\partial y} = 0 \tag{6.17}$$

$$\frac{\partial \rho u}{\partial t} + \frac{\partial \rho u^2 + p}{\partial x} + \frac{\partial \rho uv}{\partial y} = 0 \tag{6.18}$$

$$\frac{\partial \rho v}{\partial t} + \frac{\partial \rho uv}{\partial x} + \frac{\partial \rho v^2 + p}{\partial y} = 0 \tag{6.19}$$

$$\frac{\partial E}{\partial t} + \frac{\partial u(E+p)}{\partial x} + \frac{\partial v(E+p)}{\partial y} = 0. \tag{6.20}$$

Here ρ is the density of the gas, (u, v) corresponds to velocity in each component, p is the internal pressure of the gas, and E is the total energy of the gas. The total

energy is defined as

$$E = \frac{p}{\gamma - 1} + \frac{\rho}{2}\left(u^2 + v^2\right). \tag{6.21}$$

By only considering smooth wave propagation these equations may be reformulated into an equivalent non-conservative formulation. The non-conservative formulation was first studied by Hagstrom and Appelo in [22]. In their work they reformulate the equations to solve for gas volume rather than gas density, $r = 1/\rho$, thus the equations only possess quadratic nonlinearities

$$\frac{\partial r}{\partial t} + u\frac{\partial r}{\partial x} + v\frac{\partial y}{\partial y} + r\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) = 0 \tag{6.22}$$

$$\frac{\partial u}{\partial t} + v\frac{\partial u}{\partial x} + u\frac{\partial u}{\partial y} + r\frac{\partial p}{\partial x} = 0 \tag{6.23}$$

$$\frac{\partial v}{\partial t} + v\frac{\partial v}{\partial x} + u\frac{\partial v}{\partial y} + r\frac{\partial p}{\partial y} = 0 \tag{6.24}$$

$$\frac{\partial p}{\partial t} + v\frac{\partial p}{\partial x} + v\frac{\partial p}{\partial y} + \gamma p\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) = 0. \tag{6.25}$$

6.5.3 Confirming Accuracy

To validate the implementation of the scheme, I carry out the propagation of an isentropic vortex. In two dimensions the exact solution is given by

$$u = 1 - \beta \exp^{(1-r^2)} \frac{(y - y_0)}{2\pi} \tag{6.26}$$

$$v = \beta \exp^{(1-r^2)} \frac{(x - x_0)}{2\pi} \tag{6.27}$$

$$\rho = \left(1 - \left(\frac{\gamma - 1}{16\gamma\pi^2}\right)\beta^2 \exp^{2(1-r^2)}\right)^{\frac{1}{\gamma-1}} \tag{6.28}$$

$$p = \rho^\gamma. \tag{6.29}$$

	$N=1$		$N=2$		$N=3$	
h	l_∞ Error	rate	l_∞ Error	rate	l_∞ Error	rate
0.2	0.47393		0.0145243	-	0.000748291	-
0.1	0.133884	1.82	0.000587297	4.62	2.84315e-06	8.03
0.05	0.0228323	2.55	1.89461e-05	4.95	8.01525e-07	1.82

Table 6.3 : Observed point-wise errors and rates of convergence when propagating a strong vortex as governed by the Euler equations. Here an N^{th} order Dual Hermite method was used with a fourth order Runge-Kutta scheme.

Here $r^2 = (x-x_0-t)^2+(y-y_0)^2$, $\beta = 5$, $\gamma = 1$, and (x_0, y_0) denote the initial location of the Gaussian. To avoid computing analytic derivatives for the initial conditions the degrees of freedom for Hermite method are computed by projecting the function values from a Chebyshev grid to local Taylor series polynomials.

The numerical experiments are carried out over the computational domain of $[-5, 5] \times [-5, 5]$ with a Gaussian centered at the origin. The solution is propagated for a period and compared the initial solution. Table 6.3 reports the observed point-wise errors (l_∞) and Figure 6.7 illustrates the advection of the Gaussian bump.

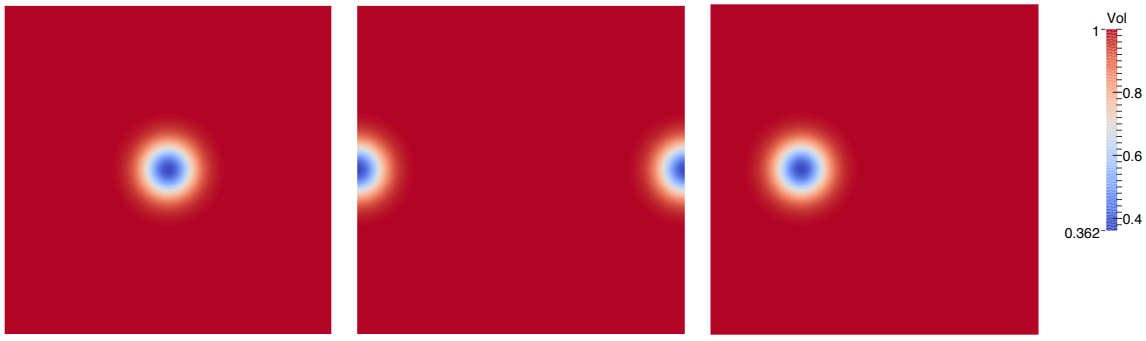


Figure 6.7 : Numerical results for advecting an isentropic vortex using the Dual Hermite method with a fourth order Runge-Kutta scheme. The domain, $[-5, 5] \times [-5, 5]$, is assumed to be periodic.

6.5.4 Convection of a Weak Vortex

In our second set of numerical experiments I consider the advection of a weak vortex. The initial conditions are given by

$$\rho = 1 - \frac{A_v^2}{2} \exp\left(1 - x^2 - y^2\right) \tag{6.30}$$

$$u = 1 - yA_v \exp\left(\frac{1 - x^2 - y^2}{2}\right) \tag{6.31}$$

$$v = 1 - xA_v \exp\left(\frac{1 - x^2 - y^2}{2}\right) \tag{6.32}$$

$$p = 1 + \frac{\lambda A_v^2}{2} \exp\left(1 - x^2 - y^2\right). \tag{6.33}$$

Figure 6.8 illustrates the propagation of a weak vortex. The domain is chosen to be $[-5, 5] \times [-5, 5]$ and the parameters are chosen to be $\lambda = 1$, and $A_v = 0.5$.

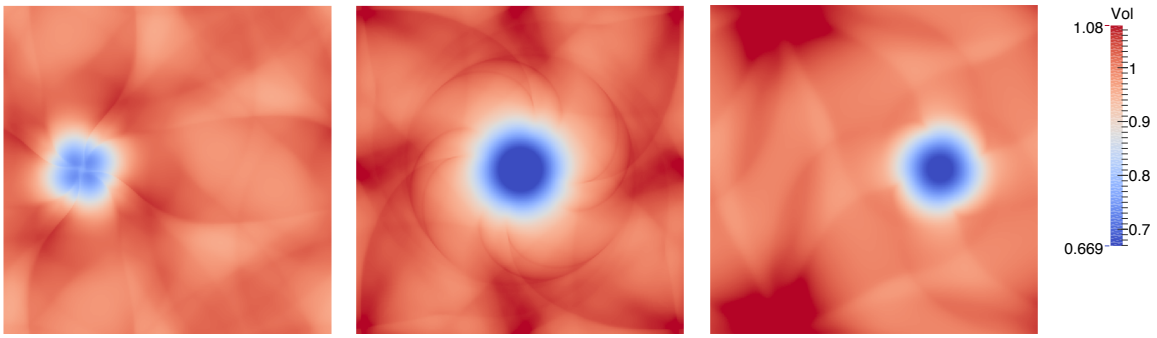


Figure 6.8 : Numerical results for advecting a weak vortex using the Dual Hermite method with a fourth order Runge-Kutta scheme. The domain, $[-5, 5] \times [-5, 5]$, is assumed to be periodic.

6.6 Summary

Building on the family of Hermite methods developed in this thesis, this chapter presents applications of those methods for simulating wave propagation. Starting

with linear wave equations, numerical experiments are presented with the elastic wave equations and the acoustic wave equations.

As modeling wave propagation in an unbounded medium is central to many applications, I present numerical experiments to aid in design choices when coupling Hermite methods with Bérenger's perfectly matched layers. Numerical experiments suggested that a quadratic function achieves comparable accuracy to a hyperbolic function. Building on the Hermite-PML framework numerical experiments are then carried out to simulate wave propagation over a synthetic velocity model.

Lastly, I discuss techniques for solving nonlinear equations within the Hermite method framework. The main bottle neck for these methods occurs when evaluating nonlinearities. In an effort to reduce the computational cost the work of Hagstrom and Appelo in [22] pioneer the use of the fast Fourier transform (FFT) to carry out right-hand side evaluations as point-wise operations. As an alternative, the doctoral work of Jang proposes a change of basis which allows for nonlinear operations to be evaluated as point-wise operations [59]. Jang's experiments had been limited to multiplicative nonlinearities but I demonstrate that these techniques may be applied to equations with other types of nonlinearities.

Chapter 7

Contributions and Future Work

The Hermite methods of Goodrich and co-authors combine Hermite interpolation and a staggered dual grid to produce high-order numerical methods [15]. These methods are relatively new given they were first introduced in 2006. They have various attractive features for scientific computing such as highly localized time-stepping, high-order approximations in both time and space, and a stability criterion independent of order. This thesis proposes extensions to the Hermite methods of Goodrich and co-authors along with algorithms which take advantage of the many core architecture of graphics processing units.

There were two main goals of this work. The first was to contribute to the development of numerical methods based on Hermite interpolation; more precisely to expand on the numerical methods proposed by Goodrich and co-authors [15]. The second goal was to tailor the algorithms to the graphics processing unit (GPU). By introducing the new extensions I was able to simplify certain aspects of Hermite methods and extend the schemes to second order equations. By employing the GPU I was able to reduce time to solution for three-dimensional simulations.

7.1 Contributions

As a starting point, Chapter 1 and 2 provided an overview of commonly used numerical methods as well as an in-depth literature review and a discussion of recent

developments in Hermite methods. Chapter 3 introduced three variations of Hermite methods which do not require a dual grid, namely the Virtual, Central, and Upwind Hermite methods. Each new numerical method is demonstrated to achieve the optimal rate of convergence of the classic Hermite method, $O(h^{2N+1})$ where h denotes the width of a cell and N is the order of the method. These methods are advantageous as they remove the need for coefficients on a dual grid. Furthermore, these methods simplify the coupling of Hermite methods and discontinuous Galerkin introduced in [19].

Chapter 4 introduced Hermite methods which use leapfrog time-stepping. The new methods may be used for either first or second order equations. Numerical experiments suggest that Hermite-Leapfrog schemes applied to second order equations converge at a rate of $O(h^{2N})$, where N is the order of the method. When the Hermite-Leapfrog scheme is applied to a first order system numerical experiments suggested rates of $O(h^{2N})$ when N is odd. When N was even a variation on rates was observed, in some cases rates of $O(h^{2N+2})$ were observed. The variation motivated the need for a truncation analysis. By following amplification errors it was observed that a Hybrid Hermite-Leapfrog scheme may be derived with $O(h^{2N+2})$ convergence rates. The Hybrid Hermite-Leapfrog scheme was derived by alternating between updates produced by the first and second order versions of the Hermite-Leapfrog methods.

To address the goal of developing high-performance algorithms, a chapter is dedicated to the use of a GPU to accelerate computations. A detailed implementation is presented along with a performance analysis. Furthermore, as storage requirements for simulations may surpass those of a single GPU, a multi-GPU implementation is described. The schemes are demonstrated to achieve linear strong scaling on three GPUs.

Lastly, as a stepping stone to future applications, Hermite methods were used to simulate linear and nonlinear wave propagation. Furthermore experiments with the split-field perfectly matched layer of Berenger were carried out in order to simulate waves in unbounded domains.

7.2 Future Work

While this thesis proposes variations on Hermite methods and demonstrates numerical stability and convergence, an immediate future direction is the development of mathematical stability arguments for the Hermite-Leapfrog schemes. At the time of writing this opportunity has spawned the collaboration between Dr. Daniel Appelö and Dr. Thomas Hagstrom. The on-going work has led to the article in progress "Globally Super-Convergent Dissipative and Conservative Hermite Methods for the Scalar Wave Equation."

A second area of interest is in expanding the applications of Hermite methods. Although these methods have various attractive features they have yet to make their way towards industrial applications. In this thesis I have begun the development of tools needed for seismic applications, for example incorporating point sources and absorbing boundary conditions. Lastly, this thesis provided scaling results on only three GPU's, an area of future work would be to study strong scaling on many more GPUs.

Appendices

Appendix A

A.1 Proof of Convergence: Virtual Hermite

Consider the following constant coefficient system,

$$u_t = Au_x, \quad A \in \mathbb{R}^{d\times d} \tag{A.1}$$
$$u\left(x,0\right) = f\left(x\right), \quad f(x+2\pi) = f(x).$$

The Virtual Hermite scheme is defined with the following operators, $\mathcal{I}$ (interpolant from primal to dual), $\tilde{\mathcal{I}}$ (interpolant from dual to primal), and the exact evolution operator $\mathcal{S}$. The following algorithm employs the following notation: p_p^k as the solution at time step k on the primal grid, and p^k as the solution on the dual grid at time step k. I emphasize that the proof of convergence equation is strictly done on the dual grid.

Algorithm 7 Virtual Hermite Algorithm

1: **procedure** Virtual Hermite Algorithm
2: Let $p^0 = \mathcal{I}f$
3: **for** $k = 0, 1, ...$ **do**
4: $p_p^k = \tilde{\mathcal{I}}p^k$
5: $p^{k+1} = \mathcal{I}\mathcal{S}p_p^k$

The convergence of the Virtual Hermite algorithm is demonstrated by the discrete evolution of the interpolant rather than the evolution of the grid data that determines the interpolants. The lemmas and theorems are those used by Goodrich et al. in

the article Hermite Methods For Hyperbolic Initial-Boundary Value Problems, see Chapter 2. As done in the Dual Hermite scheme, a technical assumption concerning the grids is made, for some fixed positive constants, c_1, c_2:

$$c_1 \max\left(h_{x_j}, h_{x_{j+1}}\right) \leq h_{x_{j+\frac{1}{2}}} \leq c_2 \min(h_{x_j}, h_{x_{j+1}}). \tag{A.2}$$

Although a dual grid is not explicitly used it is part of the analysis. Note that uniform grid not required.

Theorem A.1.1 (Virtual Hermite Convergence Theorem)**.** *Let* $\frac{\Delta t}{h_{x_j}} > 0$ *and let* $T > 0$ *be fixed. Suppose* $f \in (H^{2N+2}_{per})^d$ *and that the hypothesis of Lemma 2.2.1 holds. Then there exist a constant C, independent of* $h = \min h_{x_j}$ *so that,*

$$\|u^k - p^k\| \leq Ch^{2N+1}\|D^{2N+2}f\| \quad for \quad 0 \leq \Delta tk \leq T.$$

Proof. Consider the iterates of the Virtual Hermite algorithm,

$$p^k_p = \tilde{\mathcal{I}} p^k \tag{A.3}$$

$$p^{k+1} = \mathcal{I}\mathcal{S} p^k_p, \tag{A.4}$$

substituting the exact solution into the components yields the local truncation errors,

$$u^k = \quad \tilde{\mathcal{I}} u^k + \eta^k_p \tag{A.5}$$

$$u^{k+1} = \quad \mathcal{I}\mathcal{S} u^k + \eta^k. \tag{A.6}$$

Denote η^k_p as the error term in the first iterate, and η^k as the error from second iterate. Since evolution is exact, $u^{k+1} = \mathcal{S}u^k$ the local truncation errors are in fact

interpolation errors,

$$\eta_p^k = \quad u^k - \tilde{\mathcal{I}} u^k \tag{A.7}$$

$$\eta^k = \quad u^{k+1} - \mathcal{I} u^{k+1}. \tag{A.8}$$

Applying Lemma (A.2.1) we can provide a bound on the interpolation error for a given time step k,

$$\|\eta^k\| + \|\eta_p^k\| \leq C_k h^{2N+2} \|D^{2N+2} f\|.$$

Define $e^k = u^k - p^k$ and $e_p^k = u^k - p_p^k$. It follows that the initial interpolation error is given by,

$$\|e^0\| = \|u^0 - p^0\| = \|f - \mathcal{I} f\| \leq C_k h^{2N+2} \|D^{2N+2} f\|.$$

Subtracting the following Equation (A.5) - (A.3) and (A.6) - (A.4) provides the following error equations,

$$e_p^k = \tilde{\mathcal{I}} e^k + \eta_p^k \tag{A.9}$$

$$e^{k+1} = \mathcal{I}\mathcal{S} e_p^k + \eta^k. \tag{A.10}$$

Estimates in the Semi-Norm

We now proceed with estimates in the semi-norm, $\|\cdot\|_{N+1}$. We apply the derivative operator D^{N+1} to the error equation (A.10) and obtain the following,

$$D^{N+1} e^{k+1} = D^{N+1} \mathcal{I}\mathcal{S} e_p^k + D^{N+1} \eta^k.$$

we recall that η is the error created by interpolation and as stated by the orthogonality lemma (2.1.3), any interpolant is semi-orthogonal to any interpolation error, thus we

obtain the following equality

$$\|D^{N+1}e^{k+1}\|^2 = \|D^{N+1}\mathcal{I}\mathcal{S}e_p^k\|^2 + \|D^{N+1}\eta^k\|^2,$$

for simplicity we will denote $\|D^{N+1}\eta^k\|^2$ in terms of the order of the local truncation error,

$$\|D^{N+1}e^{k+1}\|^2 = \|D^{N+1}\mathcal{I}\mathcal{S}e_p^k\|^2 + O(h^{2N+2}). \tag{A.11}$$

Moreover we can construct the following identity,

$$D^{N+1}\mathcal{S}e_p^k = D^{N+1}\mathcal{I}\mathcal{S}e_p^k + D^{N+1}(\mathcal{S}e_p^k - \mathcal{I}\mathcal{S}e_p^k). \tag{A.12}$$

By lemma (2.1.3) we obtain the following expression,

$$\|D^{N+1}\mathcal{S}e_p^k\|^2 = \|D^{N+1}\mathcal{I}\mathcal{S}e_p^k\|^2 + \|D^{N+1}(\mathcal{S}e_p^k - \mathcal{I}\mathcal{S}e_p^k)\|^2. \tag{A.13}$$

Subtracting Equations (A.11) - (A.13) and using the fact that the $\mathcal{S}$ operator preserves the semi-norm we arrive to,

$$\|D^{N+1}e^{k+1}\|^2 - \|D^{N+1}e_p^k\|^2 = -\|D^{N+1}(\mathcal{S}e_p^k - \mathcal{I}\mathcal{S}e_p^k)\|^2 + O(h^{2N+2}). \tag{A.14}$$

We now focus on the term $\|D^{N+1}e_p^k\|^2$, and make use of Equation (A.9) and the orthogonality lemma to arrive at

$$\|D^{N+1}e_p^k\|^2 = \|D^{N+1}\tilde{\mathcal{I}}e^k\|^2 + \|D^{N+1}\eta_p^k\|^2. \tag{A.15}$$

Once again by the use of the orthogonality lemma we have

$$\|D^{N+1}\tilde{\mathcal{I}}e^k\|^2 = \|D^{N+1}e^k\|^2 - \|D^{N+1}(e^k - \tilde{\mathcal{I}}e^k)\|^2 \tag{A.16}$$

Substituting Equation (A.16) into Equation (A.15) we can arrive to an expression in terms of e^k,

$$\|D^{N+1}e_p^k\|^2 = \|D^{N+1}e^k\|^2 - \|D^{N+1}(e^k - \tilde{\mathcal{I}}e^k)\|^2 + \|D^{N+1}\eta_p^k\|^2.$$

In order to simplify we will express the error in terms of h.

$$\|D^{N+1}e_p^k\|^2 = \|D^{N+1}e^k\|^2 + O(h^{2N+2}). \tag{A.17}$$

We now combine Equation (A.17) with Equation (A.14) to arrive at the following

$$\|D^{N+1}e^{k+1}\|^2 - \|D^{N+1}e^k\|^2 = -\|D^{N+1}(\mathcal{S}e_p^k - \mathcal{I}\mathcal{S}e_p^k)\|^2 + O(h^{2N+2}). \tag{A.18}$$

Setting $\epsilon_k = \|D^{N+1}e^k\|$ and $\delta_k = \|D^{N+1}(\mathcal{S}e_p^k - \mathcal{I}\mathcal{S}e_p^k)\|$. We have the following recursive error estimate

$$\delta_k^2 = \epsilon_k^2 - \epsilon_{k+1}^2 + O(h^{2N+2}). \tag{A.19}$$

Unlike the original Hermite scheme we no longer have half-steps and arrive at the following bound,

$$\sum_{k=0}^{J} \delta_k^2 \leq \epsilon_0^2 + CJh^{2N+2} \leq Ch^{2N+1}.$$

Estimates in the L_2 Norm

With our newly established error bound we now proceed with the following L_2 esti-

mates and the conclusion of the proof.

We begin with Equation (A.10) and employ the L_2 norm,

$$\begin{aligned} \|e^{k+1}\| &\leq \|\tilde{\mathcal{I}}\mathcal{S}e_p^k\| + Ch^{2N+2} \\ &\leq \|\mathcal{S}e_p^k\| + \|\mathcal{S}e_p^k - \tilde{\mathcal{I}}\mathcal{S}e_p^k\| + Ch^{2N+2} \\ &\leq \|e_p^k\| + \|\mathcal{S}e_p^k - \tilde{\mathcal{I}}\mathcal{S}e_p^k\| + Ch^{2N+2}. \end{aligned}$$

We focus on the term $\|e_p^k\|$, employing Equation (A.9) we have the following inequality in L_2,

$$\begin{aligned} \|e_p^k\| &\leq \|\tilde{\mathcal{I}}e^k\| + Ch^{2N+2} \\ &\leq \|e^k\| + \|e^k - \tilde{\mathcal{I}}e^k\| + Ch^{2N+2} \\ &\leq \|e^k\| + Ch^{N+1}\|D^{N+1}(e^k - \tilde{\mathcal{I}}e^k)\| + Ch^{2N+2} \\ &\leq \|e^k\| + Ch^{2N+2}. \end{aligned} \tag{A.20}$$

Providing us with the inequality for the error at any time step,

$$\begin{aligned} \|e^{k+1}\| &\leq \|e^k\| + \|\mathcal{S}e_p^k - \tilde{\mathcal{I}}\mathcal{S}e_p^k\| + Ch^{2N+2} \\ &\leq \|e^k\| + Ch^{N+1}\|D^{N+1}(\mathcal{S}e_p^k - \tilde{\mathcal{I}}\mathcal{S}e_p^k)\| + Ch^{2N+2} \\ &\leq \|e^k\| + Ch^{N+1}\delta_k + Ch^{2N+2}. \end{aligned} \tag{A.21}$$

We now arrive at proposed error bound

$$\|e^{J+1}\| \leq Ch^{N+1}\sum_{k=0}^{J}\delta_k + Ch^{2N+2} \quad 0 \leq (J+1)\Delta t \leq T.$$

Finally, returning to our bound for δ_k by Cauchy-Schwarz,

$$\begin{aligned}\sum_{k=0}^{J}\delta_k &\leq C\left(\sum_{k=0}^{K}1\right)^{\frac{1}{2}}\left(\sum_{k=0}^{K}\delta_k^2\right)^{\frac{1}{2}}\\ &\leq Ch^{-\frac{1}{2}}h^{k+\frac{1}{2}}\\ &\leq Ch^k.\end{aligned}$$

Proving the theorem. □

A.2 Properties of the Hermite Interpolant

Lemma A.2.1. *The interpolation operators, $\mathcal{I}$ and $\tilde{\mathcal{I}}$, satisfy:*

$$\|g-\mathcal{I}g\| \leq Ch^{2N+2}\|D^{2N+2}g\| \quad for \quad g\in H^{2N+2}_{per} \tag{A.22}$$

$$\|D^{N+1}\left(g-\mathcal{I}g\right)\| \leq Ch^{N+1}\|D^{2N+2}g\| \quad for \quad g\in H^{2N+2}_{per} \tag{A.23}$$

$$\|g-\mathcal{I}g\| \leq Ch^{N+1}\|D^{N+1}g\| \quad for \quad g\in H^{N+1}_{per}. \tag{A.24}$$

Furthermore, as demonstrated in [15] interpolants are semi-orthogonal to interpolation error.

Lemma A.2.2. *Orthgonality Lemma*

$$(\mathcal{I}f,(g-\mathcal{I}g))_{N+1}=0 \tag{A.25}$$

Using Lemma A.2.2 we demonstrate that the interpolation operators are self-adjoint.

Lemma A.2.3. *The interpolation operators are self-adjoint*

Proof. Assume $w, q \in H^{N+1}_{per}\ \mathcal{I}$, and $\tilde{\mathcal{I}}$ direct computation shows

$$\begin{aligned}(w, \mathcal{I}q)_{N+1} = (\mathcal{I}w + (w - \mathcal{I}w), \mathcal{I}q)_{N+1} = \\ (\mathcal{I}w, \mathcal{I}q)_{N+1} + ((w - \mathcal{I}w), \mathcal{I}q)_{N+1} = \\ (\mathcal{I}w, \mathcal{I}q)_{N+1} = \\ (\mathcal{I}w, q + (\mathcal{I}q - q))_{N+1} = \\ (\mathcal{I}w, q)_{N+1} + (\mathcal{I}w, (\mathcal{I}q - q))_{N+1} = \\ (\mathcal{I}w, q)_{N+1}\,.\end{aligned}$$

□

Lemma A.2.4. *Semi-norm Lemma*

$$(f, \mathcal{I}f)_{N+1} = (\mathcal{I}f, \mathcal{I}f)_{N+1} \tag{A.26}$$

Proof. Assume $f \in H^{N+1}_{per}$ direct computation shows

$$\begin{aligned}(f, \mathcal{I}f)_{N+1} = (\mathcal{I}f + (f - \mathcal{I}f), \mathcal{I}f)_{N+1} = \\ (\mathcal{I}f, \mathcal{I}f)_{N+1} + ((f - \mathcal{I}f), \mathcal{I}f)_{N+1} = \\ (\mathcal{I}f, \mathcal{I}f)_{N+1}\,.\end{aligned}$$

□

Appendix B

B.1 High Order Point Source Approximation

In this appendix I describe how to incorporate point-source impulses for the simulation of wave propagation. Simulating wave propagation steaming from a point-source has many applications. For example, it can be used in seismic applications to model impulses from airguns which excite wave propagation within the earth's subsurface. These point-sources are typically modeled using a collection of delta functions and are a challenge for many discretizations on account of the lack of regularity in the function. This is particularly trouble sum for Hermite methods as they require smoothness in the source and initial functions.

As this is an important component of modeling wave propagation, various approaches have been developed specific to PDE discretizations. For example the work of Petersson and co-authors have developed point-source discretizations for finite difference methods by considering Fourier series expansions of the source term. These expansions result in compact support in physical space [104]. A technique that has been proposed by finite-element methods is referred to as the "total-scattered Field" formulation. In this formulation the propagating wave is assumed to be the sum of a solution which solves the homogeneous wave equation (with no forcing term) and the heterogeneous wave equation (forcing term) [105, 106]. In order to incorporate point sources into the Hermite framework, I extend the Dual Hermite methods into the Total-Scattered field framework. Although the outline is given for the Dual Her-

mite method it is general enough that it may be applied to the variations of Hermite methods presented in this thesis.

B.1.1 Inhomogeneous Acoustic Wave Equations

The inhomogeneous pressure-velocity system is expressed as

$$\frac{\partial p}{\partial t} - \nabla \cdot \mathbf{v} = f(\mathbf{x}, t) \tag{B.1a}$$

$$\frac{\partial \mathbf{v}}{\partial t} - \nabla p = 0. \tag{B.1b}$$

Here the forcing term is assumed to be of the form

$$f(\mathbf{x}, t) = f_0(t)\delta(\mathbf{x} - \mathbf{x}^*), \tag{B.2}$$

where $\mathbf{x}^*$ is the source location. The source time-function $f_0(t)$ is assumed to be sufficiently smooth and causal for all required derivatives, i.e., $f^{(n)}(t) = 0$ for all $t \leq 0$. As a starting point for the formulation I begin with the fundamental solution of the second order acoustic wave equation and then bootstrap the result to the pressure-velocity system.

B.1.2 Preliminaries in One-Dimension

In one-dimension, L will be used to denote the wave equation operator for the second order wave equation,

$$L = \frac{\partial^2}{\partial t^2} - \frac{\partial^2}{\partial x^2},$$

where we assume unit wave speed. Wave propagation in an unbounded homogeneous media with an impulsive point-source is expressed as

$$L[u] = \delta(t)\delta(x), \tag{B.3}$$

where the solution under the initial conditions $u(0,x) = 0$ and $u_t(0,x) = 0$, is given by the Green's function

$$G(x,t) = \frac{1}{2} H\left(t - |x|\right). \tag{B.4}$$

Here H denotes the heavy side function

$$H(\xi) = \begin{cases} 0; & \xi < 0 \\ \\ 1; & \xi \geq 0 \end{cases}.$$

Using the Green's function, $G(x,t)$, a fundamental solution for a general right hand side, $L[u] = f(x,t)$, may be computed by a spatial temporal convolution,

$$u(x,t) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} G(x - x', t - t') f(x', t') dx' dt'.$$

In the special case of point-sources the convolution reduces to temporal convolution only; taking $f(x,t)$ as given by Equation B.2 and applying a change of variables leads to

$$u(x,t) = \frac{1}{2} \int_0^{\infty} f_0\Big(t - t' - |x - x^*|\Big) dt'. \tag{B.5}$$

B.1.3 Solutions for First Order Equations

Equation B.5 may then be used to formulate solutions for the pressure-velocity system (Equations B.1). This is accomplished by relating both formulations. Applying a time

derivative to the first equation and divergence in the second equation leads to

$$\frac{\partial^2}{\partial t^2}p + \nabla \cdot \frac{\partial}{\partial t}\mathbf{v} = \frac{\partial}{\partial t}f,$$
$$\nabla \cdot \frac{\partial}{\partial t}\mathbf{v} + \nabla \cdot \nabla p = 0.$$

Eliminating the velocity field yields the second order system

$$\frac{\partial^2}{\partial t^2}p - \nabla^2 p = \frac{\partial}{\partial t}f.$$

Here it is observed that the source term now includes a time derivative, thus for an arbitrary point-source, the solution of the pressure field is

$$p(x,t) = \frac{1}{2}\int_0^\infty f_0'\Big(t - t' - |x - x^*|\Big)dt'.$$

Defining $\tau = t - t' - |x - x^*|$ leads to

$$p(x,t) = \frac{1}{2}\int_0^{t-|x-x^*|} f_0'(\tau)d\tau = \frac{1}{2}f_0\Big(t - |x - x^*|\Big), \tag{B.6}$$

when $t > |x - x^*|$, otherwise $p(x,t) = 0$. In order to obtain the solution for the velocity field the pressure field is differentiated in space and then integrated in time,

$$v(x,t) = \int_0^t \frac{1}{\rho}\frac{\partial}{\partial x}p(x,t')dt'.$$

From Equation B.6 we have

$$\frac{\partial}{\partial x}p(x,t) = -\frac{1}{2}\mathrm{sgn}(x - x^*)f_0'\Big(t - |x - x^*|\Big)$$

where sgn is the "sign" function,

$$\mathrm{sgn}(\xi) = \begin{cases} -1; & \xi < 0 \\ \\ 1; & \xi \geq 0 \end{cases}.$$

Lastly, we now derive the following for the velocity field

$$\begin{aligned} v(x,t) &= \int_0^t \frac{1}{2}\mathrm{sgn}(x-x^*) f_0'\Big(t' - |x-x^*|\Big) dt' \\ &= \frac{1}{2}\mathrm{sgn}(x-x^*) \int_0^{t-\frac{|x-x^*|}{c}} f_0'(\tau) d\tau \\ &= \frac{1}{2}\mathrm{sgn}(x-x^*) f_0\Big(t - |x-x^*|\Big) \end{aligned}$$

which assumes $t > \frac{|x-x^*|}{c}$. In summary for any wavelet, the fundamental solution in one-dimension is given by

$$p(x,t) = \frac{1}{2} f_0\Big(t - |x-x^*|\Big) \tag{B.7a}$$

$$v(x,t) = \frac{1}{2}\mathrm{sgn}(x-x^*) f_0\Big(t - |x-x^*|\Big) \tag{B.7b}$$

B.1.4 Total-scattered Formulation

In order to construct the Hermite total-scattered field formulation, the solution to the wave equation over a domain Ω, which will be referred to as the total wave-field, u_{tot}, is assumed to be the sum of an incident field u_{inc}, which is generated by the source,

and a scattered field u_{scat}, the propagating wave field without the contribution of the source. Clearly the solution u_{tot} solves the wave equation in the full domain Ω,

$$L[u_{tot}] = f_0(t)\delta(x - x^*), \quad x \in \Omega. \tag{B.8}$$

As the wave can be decomposed into a incident and scattered field, the linearity of the equation allows for the operator to be written as the sum of wave fields acting on the operators

$$L[u_{tot}] = L[u_{inc}] + L[u_{scat}].$$

Furthermore by partitioning the domain into a region containing a source term, Ω_{int}, and the exterior region Ω_{ext},

$$\Omega = \Omega_{int} + \Omega_{ext},$$

it can be shown the the incident and scattered field solutions satisfy the following equations

$$u_{inc} \quad \text{solves} \quad L[u_{inc}] = f_0(t)\delta(x - x^*) \quad \text{on} \quad \Omega_{int} \tag{B.9}$$

$$u_{tot} \quad \text{solves} \quad L[u_{tot}] = f_0(t)\delta(x - x^*) \quad \text{on} \quad \Omega \tag{B.10}$$

$$u_{scat} \quad \text{solves} \quad L[u_{scat}] = 0 \quad \text{on} \quad \Omega_{int}. \tag{B.11}$$

Rather than solving a PDE with a source term, the total-scattered formulation reformulates the PDE into a system of homogeneous equations and simplify enforces Equation B.11 at the interface between the decomposition of the domain. A total-

scattered field formulation solves the following PDEs

$$L[u_{scat}] = 0 \quad \text{on} \quad \Omega_{int} \tag{B.12}$$

$$L[u_{tot}] = 0 \quad \text{on} \quad \Omega_{ext} \tag{B.13}$$

$$u_{tot} = u_{scat} + u_{int} \quad \text{at the interface} \tag{B.14}$$

$$\tag{B.15}$$

B.1.5 A Total-scattered Formulation for Hermite Methods

As a first step to constructing the Hermite total-scattered field formulation it is necessary to first tag a collection of scattered nodes which will encapsulate the source term on the computational domain. Figure B.1 illustrates a one-dimensional discretization which tags a collection of nodes as scattered nodes (S) while the remaining nodes are referred to as total nodes (T). Recalling that the Dual Hermite method employs two grids, the blue nodes will correspond to nodes from the primary grid while the red nodes make up the dual grid. The circled primary node will correspond to the source of the impulse. Here solving equations B.12, and B.13 via the Dual Hermite

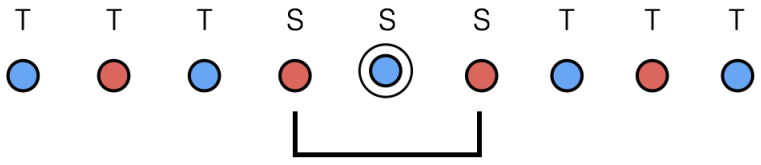


Figure B.1 : Illustration of a total-scattered field discretization for the Dual Hermite method. Blue nodes correspond to the primal nodes while the red nodes correspond to dual nodes, the source is assumed to stem from the circled node. Nodes with a T correspond to total field nodes while nodes with an S correspond to scattered field nodes.

discretization remains largely unchanged as solving the standard pressure-velocity

system with the exception that at every time step Equation B.14 must be enforced.

Introducing the impulse and enforcing Equation B.14 is accomplished by modifying the degrees of freedom so that interpolants constructed over scattered field nodes are constructed using on scattered field data. For example, Figure B.2 illustrates the interpolation procedure on the interface between the scattered and total field nodes. In order to construct the scattered field interpolant it is necessary to

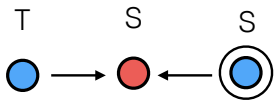


Figure B.2 : Illustration of interpolating data from subsequent primary nodes. As the interpolant is approximating data for a scattered node is becomes necessary to modify the total field node to a scatter node, this may be done by removing the contribution of the Green's function.

remove the contributions from the incident field from the total-field node. Analogously, when constructing interpolants for the total field nodes it is necessary to add the contribution from the incident field to any neighboring scattered nodes.

B.1.6 Experiments in One Dimension

To access the accuracy and estimate rates of convergence, I consider the Ricker wavelet as the impulse. By the derivativation in Section B.1.3 the analytic solution may be shown to be

$$p(x,t) = \exp\left(\omega(|x| - t)^2\right)\Big) \tag{B.16a}$$

$$v(x,t) = \operatorname{sgn}(x - x^*)\exp\left(\omega(|x| - t)^2\right)\Big). \tag{B.16b}$$

Numerical experiments are carried out on the bi-unit domain $[-1, 1]$ and the location of impulse is taken to be at the origin. The solution is propagated to a final time of $T = 1.65$ right before the wave reaches the boundary. In order to preserve accuracy in the solution it is necessary to chose the scattered field to be large enough to encompass incoming wavelet. Accuracy and observed rates of convergence are reported in Table B.1, for all numerical experiments the time step is chosen to be $\Delta t = 0.9h_x$.

	$N = 1$		$N = 2$		$N = 3$	
h	L^2 Error	rate	L^2 Error	rate	L^2 Error	rate
0.05	0.927	-	0.566	-	0.294	-
0.025	0.188	2.30	0.013	5.40	9.24e-04	8.31
0.0167	0.049	3.31	0.0010	6.19	3.08e-05	8.38
0.0125	0.017	3.69	0.00020	5.88	3.27e-06	7.79

Table B.1 : Observed convergence rates for an N order Dual Hermite method with a total scattered field formulation. Numerically the exepected rate of convergence is maintained.

B.1.7 Summary

This appendix introduced a formulation to enable modeling of one-dimensional point-sources within the Hermite method framework. The approach was constructed through the use of the total-scattered field formulation. Although these are very much preliminary results it highlights that high order accuracy may be maintained with this approach.

Bibliography

[1] P. Micikevicius, "3D finite difference computation on GPUs using CUDA," in *Proceedings of 2nd workshop on general purpose processing on graphics processing units*, pp. 79–84, ACM, 2009.

[2] X. Mei and X. Chu, "Dissecting GPU memory hierarchy through microbenchmarking," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 1, pp. 72–86, 2017.

[3] S. Čanić and E. H. Kim, "Mathematical analysis of the quasilinear effects in a hyperbolic model blood flow through compliant axi-symmetric vessels," *Mathematical Methods in the Applied Sciences*, vol. 26, no. 14, pp. 1161–1186, 2003.

[4] J. S. Hesthaven and T. Warburton, *Nodal discontinuous Galerkin methods: algorithms, analysis, and applications*. Springer Science & Business Media, 2007.

[5] M. O. Deville, P. F. Fischer, and E. H. Mund, *High-order methods for incompressible fluid flow*, vol. 9. Cambridge University Press, 2002.

[6] B. Fornberg, "The pseudospectral method: Comparisons with finite differences for the elastic wave equation," *Geophysics*, vol. 52, no. 4, pp. 483–501, 1987.

[7] J. M. Melenk and S. Sauter, "Wavenumber explicit convergence analysis for Galerkin discretizations of the Helmholtz equation," *SIAM Journal on Numerical Analysis*, vol. 49, no. 3, pp. 1210–1243, 2011.

[8] J. D. McCalpin, "A survey of memory bandwidth and machine balance in current high performance computers," *IEEE TCCA Newsletter*, vol. 19, p. 25, 1995.

[9] E. Baysal, D. D. Kosloff, and J. W. Sherwood, "Reverse time migration," *Geophysics*, vol. 48, no. 11, pp. 1514–1524, 1983.

[10] J. Virieux and S. Operto, "An overview of full-waveform inversion in exploration geophysics," *Geophysics*, vol. 74, no. 6, pp. WCC1–WCC26, 2009.

[11] A. Taflove, S. C. Hagness, *et al.*, "Computational electrodynamics: the finite-difference time-domain method," *Norwood, 2nd Edition, MA: Artech House, l995*, 1995.

[12] J. Blazek, *Computational fluid dynamics: principles and applications*. Butterworth-Heinemann, 2015.

[13] R. Gandham, D. Medina, and T. Warburton, "GPU Accelerated Discontinuous Galerkin Methods for Shallow Water Equations," *Communications in Computational Physics*, vol. 18, pp. 37–64, 7 2015.

[14] P. Courtier and J.-F. Geleyn, "A global numerical weather prediction model with variable resolution: Application to the shallow-water equations," *Quarterly Journal of the Royal Meteorological Society*, vol. 114, no. 483, pp. 1321–1346, 1988.

[15] J. Goodrich, T. Hagstrom, and J. Lorenz, "Hermite methods for hyperbolic initial-boundary value problems," *Mathematics of computation*, vol. 75, no. 254, pp. 595–630, 2006.

[16] E. T. Dye, *Performance Analysis and Optimization of Hermite Methods on NVIDIA GPUs Using CUDA*. PhD thesis, 2015.

[17] D. Appelö, M. Inkman, T. Hagstrom, and T. Colonius, "Hermite methods for aeroacoustics: Recent progress," in *17th AIAA/CEAS Aeroacoustics Conference (32nd AIAA Aeroacoustics Conference), Portland, Oregon*, 2011.

[18] D. Appelö and T. Hagstrom, "On advection by Hermite methods," *Pacific Journal Of Applied Mathematics*, vol. 4, no. 2, pp. 125–139, 2011.

[19] X. R. Chen, D. Appelö, and T. Hagstrom, "A hybrid Hermite–discontinuous Galerkin method for hyperbolic systems with application to Maxwell's equations," *Journal of Computational Physics*, vol. 257, pp. 501–520, 2014.

[20] R. Chen and T. Hagstrom, "P-adaptive Hermite methods for initial value problems," *ESAIM: Mathematical Modelling and Numerical Analysis*, vol. 46, no. 3, pp. 545–557, 2012.

[21] C. Y. Jang, D. Appelö, T. Hagstrom, and T. Colonius, "An analysis of dispersion and dissipation properties of Hermite methods and its application to direct numerical simulation of jet noise," in *18th AIAA/CEAS Aeroacoustics Conference*, 2012.

[22] T. Hagstrom and D. Appelö, "Experiments with Hermite methods for simulating compressible flows: Runge-Kutta time-stepping and absorbing layers," in *13th AIAA/CEAS Aeroacoustics Conference*, no. 2007-3505, 2007.

[23] T. Hagstrom, J. Goodrich, I. Nazarov, and C. Dodson, "High-order methods and boundary conditions for simulating subsonic flows," in *Proceedings of the 11th AIAA/CEAS Aeroacoustics Conference*, 2005.

[24] D. Medina, A. St-Cyr, and T. Warburton, "High-Order Finite-differences on multi-threaded architectures using OCCA," in *Spectral and High Order Methods for Partial Differential Equations ICOSAHOM 2014*.

[25] K. S. Yee *et al.*, "Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media," *IEEE Trans. Antennas Propag*, vol. 14, no. 3, pp. 302–307, 1966.

[26] G. D. Smith, *Numerical solution of partial differential equations: finite difference methods*. Oxford university press, 1985.

[27] B. Gustafsson, H.-O. Kreiss, and J. Oliger, *Time dependent problems and difference methods*, vol. 24. John Wiley & Sons, 1995.

[28] T. Warburton and T. Hagstrom, "Taming the CFL number for discontinuous Galerkin methods on structured meshes," *SIAM Journal on Numerical Analysis*, vol. 46, no. 6, pp. 3151–3180, 2008.

[29] D. Michéa and D. Komatitsch, "Accelerating a three-dimensional finite-difference wave propagation code using GPU graphics cards," *Geophysical Journal International*, vol. 182, no. 1, pp. 389–402, 2010.

[30] S. Brenner and R. Scott, *The mathematical theory of finite element methods*, vol. 15. Springer Science & Business Media, 2007.

[31] J. Chan, Z. Wang, A. Modave, J.-F. Remacle, and T. Warburton, "GPU-accelerated discontinuous Galerkin methods on hybrid meshes," *Journal of Computational Physics*, vol. 318, pp. 142–168, 2016.

[32] G. Cohen, *Higher-order numerical methods for transient wave equations*. Springer Science & Business Media, 2013.

[33] M. S. Gockenbach, *Understanding and implementing the finite element method*. Siam, 2006.

[34] W. H. Reed and T. Hill, "Triangular mesh methods for the neutron transport equation," *Los Alamos Report LA-UR-73-479*, 1973.

[35] J. Butcher, "General linear methods for ordinary differential equations," *Mathematics and Computers in Simulation*, vol. 79, no. 6, pp. 1834–1845, 2009.

[36] J. Chan and T. Warburton, "GPU-accelerated Bernstein-Bezier discontinuous Galerkin methods for wave problems," *arXiv preprint arXiv:1512.06025*, 2015.

[37] A. Klöckner, T. Warburton, J. Bridge, and J. S. Hesthaven, "Nodal discontinuous Galerkin methods on graphics processors," *Journal of Computational Physics*, vol. 228, no. 21, pp. 7863–7882, 2009.

[38] M. S. Fabien, *Spectral Methods for Partial Dierential Equations that Model Shallow Water Wave Phenomena*. PhD thesis, 2014.

[39] C. G. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang, "Spectral methods: Fundamentals in single domains," 2010.

[40] J. S. T. Tang, "Spectral and high-order methods with applications," 2006.

[41] B. Seibold, J.-C. Nave, and R. R. Rosales, "Jet schemes for advection problems," *arXiv preprint arXiv:1101.5374*, 2011.

[42] J.-C. Nave, R. R. Rosales, and B. Seibold, "A gradient-augmented level set method with an optimally local, coherent advection scheme," *Journal of Computational Physics*, vol. 229, no. 10, pp. 3802–3827, 2010.

[43] P. Chidyagwai, J.-C. Nave, R. R. Rosales, and B. Seibold, "A comparative study of the efficiency of jet schemes," *arXiv preprint arXiv:1104.0542*, 2011.

[44] E. Mathioudakis, N. Vilanakis, E. Papadopoulou, and Y. Saridakis, "Parallel iterative solution of the hermite collocation equations on GPUs," in *Proceedings of the World Congress on Engineering*, vol. 2, 2013.

[45] E. Mathioudakis and E. Papadopoulou, "MPI Management of Hermite Collocation Computation on a Distributed-Shared Memory System," *WSeas Transactions on Mathematics*, vol. 5, no. 5, p. 520, 2006.

[46] E. Mathioudakis, E. Papadopoulou, and Y. Saridakis, "BiCGSTAB for collocation equations on shared memory parallel architectures," *Numer. Algorithms, submitted*, 2001.

[47] G. Dahlquist and Å. Björck, "Numerical methods in scientific computing, volume i," *SIAM, Philadelphia*, vol. 55, 2007.

[48] C. De Boor, K. Höllig, and M. Sabin, "High accuracy geometric hermite interpolation," *Computer Aided Geometric Design*, vol. 4, no. 4, pp. 269–278, 1987.

[49] P. G. Ciarlet and P. Raviart, "General lagrange and hermite interpolation in rn with applications to finite element methods," *Archive for Rational Mechanics and Analysis*, vol. 46, no. 3, pp. 177–199, 1972.

[50] G. Birkhoff, M. H. Schultz, and R. S. Varga, "Piecewise Hermite interpolation in one and two variables with applications to partial differential equations," *Numerische Mathematik*, vol. 11, no. 3, pp. 232–256, 1968.

[51] A. Harten, B. Engquist, S. Osher, and S. R. Chakravarthy, "Uniformly high order accurate essentially non-oscillatory schemes, III," *Journal of Computational Physics*, vol. 131, no. 1, pp. 3–47, 1997.

[52] A. Kornelus and D. Appelö, "Flux-conservative hermite methods for simulation of nonlinear conservation laws," *arXiv preprint arXiv:1703.06848*, 2017.

[53] A. Kornelus and D. Appelö, "On the scaling of entropy viscosity in high order methods," *arXiv preprint arXiv:1703.06917*, 2017.

[54] A. Kornelus and D. Appelö, "Application of the entropy viscosity method to Hermite methods for shock problems,"

[55] J.-L. Guermond and R. Pasquetti, "Entropy viscosity method for high-order approximations of conservation laws,"

[56] R. J. LeVeque, *Numerical methods for conservation laws*. Springer Science & Business Media, 1992.

[57] A. Klöckner, T. Warburton, and J. S. Hesthaven, "Viscous shock capturing in a time-explicit discontinuous Galerkin method," *Mathematical Modelling of Natural Phenomena*, vol. 6, no. 3, pp. 57–83, 2011.

[58] T. A. Driscoll, N. Hale, and L. N. Trefethen, *Chebfun Guide*. Pafnuty Publications, 2014.

[59] C. Y. Jang, *Contributions to the theory and applications of Hermite methods*. PhD thesis, Southern Methodist University, 2015.

[60] R. Courant, E. Isaacson, and M. Rees, "On the solution of nonlinear hyperbolic differential equations by finite differences," *Communications on Pure and Applied Mathematics*, vol. 5, no. 3, pp. 243–255, 1952.

[61] P. Lesaint and P.-A. Raviart, "On a finite element method for solving the neutron transport equation," *Mathematical aspects of finite elements in partial differential equations*, no. 33, pp. 89–123, 1974.

[62] J. Qiu and C.-W. Shu, "On the construction, comparison, and local characteristic decomposition for high-order central WENO schemes," *Journal of Computational Physics*, vol. 183, no. 1, pp. 187–209, 2002.

[63] Y.-X. Ren, H. Zhang, *et al.*, "A characteristic-wise hybrid compact-WENO scheme for solving hyperbolic conservation laws," *Journal of Computational Physics*, vol. 192, no. 2, pp. 365–386, 2003.

[64] B. Cockburn and C.-W. Shu, "The local discontinuous Galerkin method for time-dependent convection-diffusion systems," *SIAM Journal on Numerical Analysis*, vol. 35, no. 6, pp. 2440–2463, 1998.

[65] K. Roberts and N. Weiss, "Convective difference schemes," *Mathematics of Computation*, vol. 20, no. 94, pp. 272–299, 1966.

[66] T. Hagstrom and D. Appelö, "Solving PDEs with Hermite Interpolation," in *Spectral and High Order Methods for Partial Differential Equations ICOSAHOM 2014*, pp. 31–49, Springer, 2015.

[67] C. K. Tam and J. C. Webb, "Dispersion-relation-preserving finite difference schemes for computational acoustics," *Journal of computational physics*, vol. 107, no. 2, pp. 262–281, 1993.

[68] O. C. Zienkiewicz and J. Z. Zhu, "The superconvergent patch recovery and a posteriori error estimates. Part 1: The recovery technique," *International Journal for Numerical Methods in Engineering*, vol. 33, no. 7, pp. 1331–1364, 1992.

[69] B. Cockburn, B. Dong, and J. Guzmán, "Optimal convergence of the original DG method for the transport-reaction equation on special meshes," *SIAM Journal on Numerical Analysis*, vol. 46, no. 3, pp. 1250–1265, 2008.

[70] R. LeVeque, *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems (Classics in Applied Mathematics Classics in Applied Mathematics)*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2007.

[71] A. Greenbaum and T. P. Chartier, *Numerical methods: design, analysis, and computer implementation of algorithms*. Princeton University Press, 2012.

[72] A. Iserles, "Generalized leapfrog methods," *IMA Journal of Numerical Analysis*, vol. 6, no. 4, pp. 381–392, 1986.

[73] S.-C. Yang, Z. Chen, Y.-q. Yu, and W.-Y. Yin, "The unconditionally stable one-step leapfrog ADI-FDTD method and its comparisons with other FDTD methods," *Microwave and Wireless Components Letters, IEEE*, vol. 21, no. 12, pp. 640–642, 2011.

[74] B. J. Davis, *A study into discontinuous Galerkin methods for the second order wave equation*. PhD thesis, Monterey, California: Naval Postgraduate School, 2015.

[75] D. Appelo and T. Hagstrom, "A new discontinuous Galerkin formulation for wave equations in second-order form," *SIAM Journal on Numerical Analysis*, vol. 53, no. 6, pp. 2705–2726, 2015.

[76] A. Modave, A. St-Cyr, W. A. Mulder, and T. Warburton, "A nodal discontinuous Galerkin method for reverse-time migration on GPU clusters," *Geophysical Journal International*, vol. 203, no. 2, pp. 1419–1435, 2015.

[77] "Discontinous galerkin and finite difference methods for the acoustic equations with smooth coefficients," Master's thesis.

[78] W. W. Symes, "Reverse time migration with optimal checkpointing," *Geophysics*, vol. 72, no. 5, pp. SM213–SM221, 2007.

[79] P. Channell and C. Scovel, "Symplectic integration of Hamiltonian systems," *Nonlinearity*, vol. 3, no. 2, p. 231, 1990.

[80] D. Medina, "OKL: A Unified Language for Parallel Architectures," 2015.

[81] J. Sanders and E. Kandrot, *CUDA by Example: An Introduction to General-Purpose GPU Programming*. Addison-Wesley Professional, 1st ed., 2010.

[82] J. E. Stone, D. Gohara, and G. Shi, "OpenCL: A parallel programming standard for heterogeneous computing systems," *Computing in science & engineering*, vol. 12, no. 1-3, pp. 66–73, 2010.

[83] P. Pacheco, *An Introduction to Parallel Programming*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1st ed., 2011.

[84] R. Hornung, J. Keasler, *et al.*, "The RAJA portability layer: overview and status," *Lawrence Livermore National Laboratory, Livermore, USA*, 2014.

[85] H. C. Edwards and C. R. Trott, "Kokkos: Enabling performance portability across manycore architectures," in *Extreme Scaling Workshop (XSW), 2013*, pp. 18–24, IEEE, 2013.

[86] S. Williams, A. Waterman, and D. Patterson, "Roofline: an insightful visual performance model for multicore architectures,"

[87] K. J. Marfurt, "Accuracy of finite-difference and finite-element modeling of the scalar and elastic wave equations," *Geophysics*, vol. 49, no. 5, pp. 533–549, 1984.

[88] D. Appelö and N. A. Petersson, "A stable finite difference method for the elastic wave equation on complex geometries with free surfaces," *Communications in Computational Physics*, vol. 5, no. 1, pp. 84–107, 2009.

[89] J. Chan, "Weight-adjusted discontinuous Galerkin methods: matrix-valued weights and elastic wave propagation in heterogeneous media," *arXiv preprint arXiv:1701.00215*, 2017.

[90] R. L. Higdon, "Numerical absorbing boundary conditions for the wave equation," *Mathematics of computation*, vol. 49, no. 179, pp. 65–90, 1987.

[91] R. L. Higdon, "Absorbing boundary conditions for difference approximations to the multidimensional wave equation," *Mathematics of computation*, vol. 47, no. 176, pp. 437–459, 1986.

[92] P. G. Petropoulos, L. Zhao, and A. C. Cangellaris, "A reflectionless sponge layer absorbing boundary condition for the solution of Maxwell's equations with high-order staggered finite difference schemes," *Journal of Computational Physics*, vol. 139, no. 1, pp. 184–208, 1998.

[93] J. LaGrone and T. Hagstrom, "Double absorbing boundaries for finite-difference time-domain electromagnetics," *Journal of Computational Physics*, vol. 326, pp. 650 – 665, 2016.

[94] M. Israeli and S. A. Orszag, "Approximation of radiation boundary conditions," *Journal of Computational Physics*, vol. 41, no. 1, pp. 115–135, 1981.

[95] T. Colonius and H. Ran, "A super-grid-scale model for simulating compressible flow on unbounded domains," *Journal of Computational Physics*, vol. 182, no. 1, pp. 191–212, 2002.

[96] J.-P. Berenger, "A perfectly matched layer for the absorption of electromagnetic waves," *Journal of computational physics*, vol. 114, no. 2, pp. 185–200, 1994.

[97] K. C. Meza-Fajardo and A. S. Papageorgiou, "A nonconvolutional, split-field, perfectly matched layer for wave propagation in isotropic and anisotropic elastic media: stability analysis," *Bulletin of the Seismological Society of America*, vol. 98, no. 4, pp. 1811–1836, 2008.

[98] Q.-H. Liu and J. Tao, "The perfectly matched layer for acoustic waves in absorptive media," *The Journal of the Acoustical Society of America*, vol. 102, no. 4, pp. 2072–2082, 1997.

[99] D. Komatitsch and J. Tromp, "A perfectly matched layer absorbing boundary condition for the second-order seismic wave equation," *Geophysical Journal*

International, vol. 154, no. 1, pp. 146–153, 2003.

[100] P. G. Petropoulos, "On the termination of the perfectly matched layer with local absorbing boundary conditions," *Journal of Computational Physics*, vol. 143, no. 2, pp. 665–673, 1998.

[101] A. Modave, *Absorbing Layers for Wave-Like Time-Dependent Problems-Design, Discretization and Optimization.* PhD thesis, Université de Liège, Liège, Belgique, 2013.

[102] A. Bermúdez, L. Hervella-Nieto, A. Prieto, and R. Rodríguez, "An exact bounded perfectly matched layer for time-harmonic scattering problems," *SIAM Journal on Scientific Computing*, vol. 30, no. 1, pp. 312–338, 2007.

[103] C. Puelz, B. Riviere, S. Canic, and C. G. Rusin, "Comparison of reduced models for blood flow using Runge-Kutta discontinuous Galerkin methods," *arXiv preprint arXiv:1511.05277*, 2015.

[104] N. A. Petersson, O. O'Reilly, B. Sjögreen, and S. Bydlon, "Discretizing singular point sources in hyperbolic wave propagation problems," *Journal of Computational Physics*, vol. 321, pp. 532–555, 2016.

[105] T. R. Atcheson, "Accelerated Plane-wave Discontinuous Galerkin for Heterogeneous Scattering Problems," 2015.

[106] R. Hiptmair, A. Moiola, and I. Perugia, "A survey of Trefftz methods for the Helmholtz equation," *arXiv preprint arXiv:1506.04521*, 2015.